

Stochastic Block-Coordinate Frank-Wolfe Optimization for Structural SVMs

Simon Lacoste-Julien*

INRIA - SIERRA project-team

École Normale Supérieure, Paris, France

Martin Jaggi*

CMAP

École Polytechnique, Palaiseau, France

Mark Schmidt

INRIA - SIERRA project-team

École Normale Supérieure, Paris, France

Patrick Pletscher

Machine Learning Laboratory

ETH Zurich, Switzerland

Abstract

We consider the use of Frank-Wolfe optimization algorithms on the dual formulation of structural SVMs. These yield simple algorithms which only need access to an approximate maximization oracle for the structured prediction problem and thus have wide applicability. This perspective provides insights on previous popular algorithms as we show that batch subgradient as well as the cutting plane algorithms are equivalent to versions of Frank-Wolfe algorithms, enabling us to improve on their convergence analysis by harvesting the Frank-Wolfe literature. Moreover, we propose a new stochastic coordinate descent version of Frank-Wolfe which yields a provably convergent optimization algorithm for structural SVMs with total run-time independent of the number of training examples, like Pegasos, but with duality gap certificate guarantees and step-size robustness thanks to the use of line-search. Our experiments on sequence prediction indicate that this simple algorithm outperforms all other optimization algorithms which only have access to the maximization oracle.

1. Introduction

The popularity of binary SVMs as a general classification toolbox has attracted interest in recent years for tailored convex optimization solvers applied to its large margin learning objective. The choice has been more limited however for the extension of SVMs to structured outputs (such as graphs and other combinatorial objects (Taskar et al., 2003; Tsochantaridis et al., 2005)), probably due to the additional complexity of dealing with an exponential number of variables/constraints, and the diversity of ways to exploit their structure. On the other hand, one of the oldest (and simplest) constrained optimization algorithms, the Frank-Wolfe algorithm (Frank and Wolfe, 1956) (also called *conditional gradient* (Bertsekas, 1999)), has seen a surge of interest recently, both in machine learning and signal processing in general (Mangasarian, 1995; Clarkson, 2010; Jaggi, 2011; Bach, 2011), and binary SVMs in particular (Keerthi et al., 2000; Gärtner and Jaggi, 2009; Ouyang and Gray, 2010). This is enabled by useful properties such as requiring only the efficient optimization of linear functions on (possibly large) constraint sets, as well as yielding

*Both authors contributed equally.

sparse iterates. These properties make it a natural candidate optimizer for the differentiable dual objective of the structural SVM formulation.

In this paper, we thus consider the use of Frank-Wolfe optimization algorithms for structural SVMs (Taskar et al., 2003; Tsochantaridis et al., 2005). Despite the exponential number of variables to consider, the linear optimization over the constraint set turns out to be equivalent to the so-called *loss-augmented decoding* subproblem (hereafter called a *maximization oracle*) which can often be solved efficiently and is used by other popular structural SVM optimization algorithms such as for the subgradient algorithms (Ratliff et al., 2007; Shalev-Shwartz et al., 2010) or the cutting plane approaches (Joachims et al., 2009; Teo et al., 2010). In this paper, we will focus on optimizers which only require such maximization oracles, as these give the widest applicability for structured prediction. Such oracles exist (sometimes in approximate form) for a wide range of structures, such as the dynamic programming structure coming from a graphical model formulation on the labels (Taskar et al., 2003), as well as other combinatorial objects such as graph matchings (Caetano et al., 2009) or associative Markov networks (Taskar, 2004). In contrast, other approaches make use of more expensive oracles, such as doing marginal inference on a graphical model defined on the labels (called the *expectation* oracle in the summary Table 1), or doing a Bregman projection on the space of structures (Taskar et al., 2006), though these operations are generally less efficient than maximization oracles. We can also distinguish between batch algorithms which update the parameters only after processing all the training data, and online algorithms which update after every datapoint such as stochastic subgradient methods (Ratliff et al., 2007; Shalev-Shwartz et al., 2010) or the online exponentiated gradient approach (Collins et al., 2008). We summarize a few of the most popular algorithms in Table 1 with their convergence rates quoted in number of oracle calls to reach an accuracy of ε in terms of the relevant quantities defined in Section 2.

By considering the Frank-Wolfe perspective, we make the following contributions in this paper:

- We show that the batch Frank-Wolfe algorithm on the structural SVM dual objective is equivalent to batch subgradient descent in the primal, suggesting a new line-search version of the subgradient method, as well as improving its convergence rate over the analysis provided in Shalev-Shwartz et al. (2010). We also show that the min-norm-point extension of Frank-Wolfe, which in each iteration re-optimizes over all previously visited coordinates (Clarkson, 2010), is equivalent to the cutting plane algorithm of SVM-Struct (Joachims et al., 2009). Because of recent advances giving duality gap convergence rates for Frank-Wolfe algorithms with ε -approximate oracles, we obtain new primal rate guarantees for the batch subgradient and cutting plane algorithms even with approximate oracles.
- We propose a new stochastic coordinate descent version of Frank-Wolfe on product domains with similar provable convergence rates. By applying it to structural SVMs, we obtain an online algorithm where the number of required oracle calls to reach a precision ε is independent from the number of training examples (see Table 1), where only ε -approximate maximization oracles are required, and which can provide a duality gap certificate.
- Our experimental results on sequence prediction match closely the theoretical analysis: the line search yields a significant advantage in the first few passes compared to the stochastic subgradient approach of Pegasos (Shalev-Shwartz et al., 2010), and there is a systematic (but smaller) advantage in the later passes due to the difference of a logarithmic factor in the rates.

2. Problem Setup: Large Margin Structured Prediction

We briefly review the standard convex optimization setup for large margin learning for structured prediction (Taskar et al., 2003; Tsochantaridis et al., 2005). In structured prediction, the goal is

Optimization algorithm	Primal/Dual	Type of guarantee	Oracle type	Convergence rate
dual extragradient (Taskar et al., 2006)	primal-‘dual’	saddle point gap	Bregman projection	$O\left(\frac{nR \log \mathcal{Y} }{\lambda \varepsilon}\right)$
online exponentiated gradient (Collins et al., 2008)	dual	expected dual error	expectation	$O\left(\frac{nR^2 \log \mathcal{Y} }{\lambda \varepsilon}\right)$
excessive gap reduction (Zhang et al., 2011)	primal-dual	duality gap	expectation	$O\left(nR \sqrt{\frac{\log \mathcal{Y} }{\lambda \varepsilon}}\right)$
BMRM (Teo et al., 2010)	primal	\geq primal error	maximization	$O\left(\frac{nR^2}{\lambda \varepsilon}\right)$
1-Slack SVM-Struct (Joachims et al., 2009)	primal-dual	duality gap	maximization	$O\left(\frac{nR^2}{\lambda \varepsilon}\right)$
Pegasos (Shalev-Shwartz et al., 2010)	primal	primal error w.h.p.	maximization	$\tilde{O}\left(\frac{R^2}{\lambda \varepsilon}\right)$
this paper: stochastic coordinate descent Frank-Wolfe	primal-dual	expected duality gap	maximization	$O\left(\frac{R^2}{\lambda \varepsilon}\right)$ Thm. 3

Table 1: Convergence rates given in the *number of calls to the oracles* for different optimization algorithms for the structural SVM objective (1) in the case of a Markov random field structure, to reach a specific accuracy ε measured for different types of gaps, in term of the number of training examples n , regularization parameter λ , size of the label space $|\mathcal{Y}|$, maximum feature norm $R := \max_{i, \mathbf{y}} \|\psi_i(\mathbf{y})\|_2$ (some minor terms were ignored for succinctness). The \tilde{O} notation ignores the logarithmic terms – this appears in the case of Pegasos as its bound on the error decreases as $\log(k)/k$, where k is the number of iterations. Notice that only Pegasos and our proposed algorithm have rates independent of n .

to predict a structured object $\mathbf{y} \in \mathcal{Y}(\mathbf{x})$ (such as a sequence of tags) for a given input $\mathbf{x} \in \mathcal{X}$. In the standard approach, a structured feature map $\phi : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{H}$ encoding the relevant information for input/output pairs is defined in such a way so that a linear classifier $h_{\mathbf{w}}$ with parameter \mathbf{w} and taking the form $h_{\mathbf{w}}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle$ can be computed efficiently (for example using dynamic programming). Given a labelled training set $\mathcal{D} = \{(\mathbf{x}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}})\}_{i=1}^n$, \mathbf{w} is learned by solving the following optimization problem which encodes the large margin criterion for structural SVM (Tsochantaridis et al., 2005; Taskar et al., 2003):

$$\begin{aligned}
\min_{\mathbf{w}, \xi} \quad & \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^n \xi_i \\
\text{s.t.} \quad & \langle \mathbf{w}, \underbrace{\phi(\mathbf{x}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}}) - \phi(\mathbf{x}_i^{\mathcal{D}}, \mathbf{y})}_{=: \psi_i(\mathbf{y})} \rangle \geq \underbrace{L(\mathbf{y}_i^{\mathcal{D}}, \mathbf{y})}_{=: L_i(\mathbf{y})} - \xi_i \quad \forall i \in [n], \forall \mathbf{y} \in \underbrace{\mathcal{Y}(\mathbf{x}_i^{\mathcal{D}})}_{=: \mathcal{Y}_i}.
\end{aligned} \tag{1}$$

$L_i(\mathbf{y}) := L(\mathbf{y}_i^{\mathcal{D}}, \mathbf{y})$ denotes the task-dependent structured error of predicting output \mathbf{y} instead of the observed output $\mathbf{y}_i^{\mathcal{D}}$ (a Hamming distance between the two labels for example). ξ_i is a slack variable measuring the surrogate loss for the i -th data point (how well the margin is satisfied) and λ is the regularization parameter. The convex problem in (1) is what Joachims et al. (2009, Optimization Problem 2) calls the n -slack structural SVM with margin-rescaling. Besides the *margin-rescaled* formulation above, (Tsochantaridis et al., 2005) also proposed a *slack-rescaled* variant, which we note can be obtained by replacing $\psi_i(\mathbf{y})$ in (1) by $\psi_i^{SR}(\mathbf{y}) := L_i(\mathbf{y})\psi_i(\mathbf{y})$.

Non-Smooth Formulation and Loss-Augmented Decoding For structured prediction, the above problem can have an exponential number of constraints due to the combinatorial nature of \mathcal{Y} . We can replace the $\sum_i |\mathcal{Y}_i|$ linear constraints with n *non-linear* ones by defining the structured hinge-loss:

$$\tilde{H}_i(\mathbf{w}) := \max_{\mathbf{y} \in \mathcal{Y}_i} \underbrace{L_i(\mathbf{y}) - \langle \mathbf{w}, \psi_i(\mathbf{y}) \rangle}_{=: H_i(\mathbf{y}; \mathbf{w})}. \tag{2}$$

The constraints in (1) can thus be replaced with the non-linear ones $\xi_i \geq \tilde{H}_i(\mathbf{w})$. The computation of the structured hinge-loss for each i amounts to finding the most “violating” output \mathbf{y} for a

given input \mathbf{x}_i , a task which can be carried out efficiently in typical structured prediction settings (under suitable assumptions of decomposability of the feature map ϕ and the error function L). This problem is called the *loss-augmented decoding* subproblem. In this paper, we only assume access to an efficient solver for this subproblem (the maximization oracle). Because the sum of ξ_i is minimized in (1), the constraint $\xi_i \geq \tilde{H}_i(\mathbf{w})$ is tight at the optimum and so an equivalent non-smooth unconstrained formulation of (1) is:

$$\min_{\mathbf{w}} \quad \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^n \tilde{H}_i(\mathbf{w}). \quad (3)$$

An obvious algorithm to optimize this formulation is to use subgradient descent, such as in Ratliff et al. (2007). A subgradient of $\tilde{H}_i(\mathbf{w})$ with respect to \mathbf{w} is easily found as $-\psi_i(\mathbf{y}_i^*)$ for \mathbf{y}_i^* being a maximizer in the loss-augmented decoding subproblem (2). In the slack-rescaled variant of the structural SVM, the loss-augmented subproblem (2) with $\psi_i^{SR}(\mathbf{y})$ in place of $\psi_i(\mathbf{y})$ generally becomes more difficult to solve than in the margin-rescaled variant. For this reason we here focus on the margin-rescaled version, however all the analysis also applies to the slack-rescaled case.

The Dual of the n-Slack-Formulation The Lagrange dual problem of the above n -slack-formulation (1) has $m := \sum_i |\mathcal{Y}_i|$ many variables or potential ‘support vectors’. It is given by:

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^m} \quad f(\alpha) &:= \frac{\lambda}{2} \left\| \underbrace{\sum_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \alpha_i(\mathbf{y}) \frac{\psi_i(\mathbf{y})}{\lambda n}}_{=: \mathbf{w} = A\alpha} \right\|^2 - \underbrace{\sum_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \alpha_i(\mathbf{y}) \frac{L_i(\mathbf{y})}{n}}_{=: \mathbf{b}^T \alpha} \\ \text{s.t.} \quad &\sum_{\mathbf{y} \in \mathcal{Y}_i} \alpha_i(\mathbf{y}) = 1 \quad \forall i \in [n] \quad \text{and} \quad \alpha_i(\mathbf{y}) \geq 0 \quad \forall i \in [n], \forall \mathbf{y} \in \mathcal{Y}_i. \end{aligned} \quad (4)$$

We denote by $\alpha_i(\mathbf{y})$ the dual variable associated to the training example i and potential output $\mathbf{y} \in \mathcal{Y}_i$. In this work, for some given dual variable vector α , we will often consider the corresponding primal variable $\mathbf{w} = \sum_{i, \mathbf{y} \in \mathcal{Y}_i} \alpha_i(\mathbf{y}) \frac{\psi_i(\mathbf{y})}{\lambda n}$ obtained from the KKT condition which needs to hold at optimality of the two above convex optimization problems – see Appendix B.

To simplify notation, we introduce the matrix $A \in \mathbb{R}^{d \times m}$ consisting of the m many columns $A = \{ \frac{1}{\lambda n} \psi_i(\mathbf{y}) \in \mathbb{R}^d \mid i \in [n], \mathbf{y} \in \mathcal{Y}_i \}$. Using this, our primal-dual correspondence between \mathbf{w} and α simply writes as $\mathbf{w} = A\alpha$. Also, our dual optimization objective (4) simplifies to $f(\alpha) := \frac{\lambda}{2} \|A\alpha\|^2 - \mathbf{b}^T \alpha$ for the fixed vector $\mathbf{b} \in \mathbb{R}^m$ s.t. $\mathbf{b} := (\frac{1}{n} L_i(\mathbf{y}))_{i \in [n], \mathbf{y} \in \mathcal{Y}_i}$. Here the domain $\mathcal{M} \subset \mathbb{R}^m$ is the product of n simplices, $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$. Since all methods considered in this paper are first-order optimization algorithms, the gradient $\nabla f(\alpha) = \lambda A^T A\alpha - \mathbf{b} = \lambda A^T \mathbf{w} - \mathbf{b}$ of our (dual) objective function $f(\alpha)$ will be a crucial quantity.

3. Frank-Wolfe Algorithms for Constrained Convex Optimization

In this section, we review the Frank-Wolfe algorithm and present a new general coordinate-wise version which is of interest for general constrained optimization over product domains. We describe how to apply Frank-Wolfe on the dual of the structural SVM in Section 4 and the coordinate-wise version in Section 5.

The Frank-Wolfe Algorithm We consider the convex optimization problem $\min_{\alpha \in \mathcal{M}} f(\alpha)$, where the convex feasible set \mathcal{M} is *compact* and the convex objective f is *continuously differentiable*. The Frank-Wolfe algorithm (Frank and Wolfe, 1956) (listed in Algorithm 1) is an iterative optimization algorithm for such problems that only requires to be able to optimize *linear* functions over \mathcal{M} , and thus has wide applicability. At every iteration, a feasible search corner \mathbf{s} is first found by minimizing over \mathcal{M} the *linearization* of f at the current iterate α (see picture in inset).

Algorithm 1: Frank-Wolfe on a Compact Convex Domain

Let $\alpha^{(0)} \in \mathcal{M}$
for $k = 0 \dots K$ **do**
 Let $\gamma := \frac{2}{k+2}$
 Compute $\mathbf{s} := \operatorname{argmin}_{\mathbf{s}' \in \mathcal{M}} \langle \mathbf{s}', \nabla f(\alpha^{(k)}) \rangle$
 Optionally: Choose γ by line-search
 Update $\alpha^{(k+1)} := (1 - \gamma)\alpha^{(k)} + \gamma\mathbf{s}$
end

Algorithm 2: Coordinate Descent with “Frank-Wolfe”-Type Updates on Product Domain

Let $\alpha^{(0)} \in \mathcal{M} = \mathcal{M}^{(1)} \times \dots \times \mathcal{M}^{(n)}$
for $k = 0 \dots K$ **do**
 Pick $i \in_{u.a.r.} [n]$
 Let $\gamma := \frac{2n}{k+2n}$
 Find $\mathbf{s}_{(i)} := \operatorname{argmin}_{\mathbf{s}'_{(i)} \in \mathcal{M}^{(i)}} \langle \mathbf{s}'_{(i)}, \nabla_{(i)} f(\alpha^{(k)}) \rangle$
 Optionally: Choose γ by line-search
 Update $\alpha_{(i)}^{(k+1)} := \alpha_{(i)}^{(k)} + \gamma(\mathbf{s}_{(i)} - \alpha_{(i)}^{(k)})$
end

The next iterate is then obtained as a convex combination of \mathbf{s} and the previous iterate, with step-size γ . These simple updates yield two additional interesting properties for this algorithm. First, every iterate $\alpha^{(k)}$ can be written as a convex combination of the starting point $\alpha^{(0)}$ and the search corners \mathbf{s} found previously. $\alpha^{(k)}$ thus has a sparse representation, which makes the algorithm suitable even for cases where the dimensionality of α is huge. Second, since f is convex, the minimum of the linearization of f over \mathcal{M} immediately gives a lower bound on the value of the yet unknown optimal solution $f(\alpha^*)$. Every step of the algorithm thus computes for free the following “linearization duality gap” defined for any feasible point $\alpha \in \mathcal{M}$ (which is in fact a special case of the Fenchel duality gap as explained in Appendix A):

$$g(\alpha) := \max_{\mathbf{s}' \in \mathcal{M}} \langle \alpha - \mathbf{s}', \nabla f(\alpha) \rangle = \langle \alpha - \mathbf{s}, \nabla f(\alpha) \rangle. \quad (5)$$

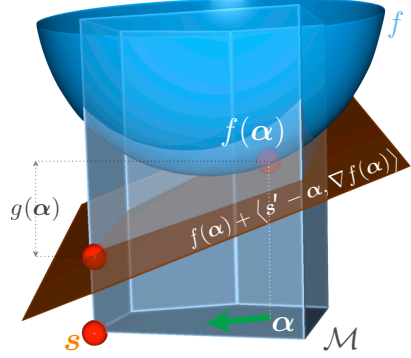
As $g(\alpha) \geq f(\alpha) - f(\alpha^*)$ by the above argument, \mathbf{s} thus readily gives at each iteration the current duality gap as a *certificate* for the current approximation quality¹ allowing us to monitor the convergence, and more importantly to choose the theoretically sound stopping criterion $g(\alpha^{(k)}) \leq \varepsilon$, instead of specifying a maximum iterate K .

In terms of convergence, it is known that, after $O(\frac{1}{\varepsilon})$ many iterations, Algorithm 1 obtains an ε -approximate solution (Frank and Wolfe, 1956; Dunn and Harshbarger, 1978), and a guaranteed ε -small duality gap (Clarkson, 2010; Jaggi, 2011), along with a certificate \mathbf{s} . For the convergence results to hold, the internal linear subproblem must not necessarily be solved exactly, but only to some additive error, as we will briefly discuss in Section 6. We will generalize and review the convergence proof in Appendix E. The constant hidden in the $O(\frac{1}{\varepsilon})$ notation is the *curvature constant* C_f (alternatively also called the *strong smoothness* constant of f), which is essentially the Lipschitz-constant of the gradient ∇f , times the squared diameter of \mathcal{M} , see e.g. our Appendix C for a formal definition.

Block-Coordinate Frank-Wolfe Algorithm for Product Domains Algorithm 2 represents the main new optimization contribution of this paper, being a provably convergent block-coordinate version of the Frank-Wolfe algorithm for constrained convex optimization problems

$$\min_{\alpha \in \mathcal{M}^{(1)} \times \dots \times \mathcal{M}^{(n)}} f(\alpha) \quad (6)$$

¹See also Jaggi (2011).



where the domain has the structure of a Cartesian product $\mathcal{M} = \mathcal{M}^{(1)} \times \dots \times \mathcal{M}^{(n)} \subseteq \mathbb{R}^m$ over n many factors, or blocks. The main idea of the method is to perform cheaper update steps that only affect a single variable block $\mathcal{M}^{(i)}$, and not all of them simultaneously. This is motivated by coordinate descent methods, which have a very successful history when applied to large scale optimization. Here we assume that each factor $\mathcal{M}^{(i)} \subseteq \mathbb{R}^{m_i}$ is convex and *compact*, and $\sum_{i=1}^n m_i = m$. We will write $\alpha_{(i)} \in \mathbb{R}^{m_i}$ for the i -th block of coordinates of a vector $\alpha \in \mathbb{R}^m$.

Algorithm 2 picks a single one of the n factors uniformly at random, and in each step leaves all other factors unchanged. If there is only one factor ($n = 1$), then Algorithm 2 becomes the standard Frank-Wolfe Algorithm 1. The algorithm can be interpreted as a simplification of Nesterov’s “huge-scale” uniform coordinate descent method (Nesterov, 2010, Section 4). Here, instead of solving a more complicated proximal operator, we only need to solve a linear subproblem in each iteration.

Convergence Results The following main theorem shows that after $O(\frac{1}{\varepsilon})$ many iterations, Algorithm 2 obtains an ε -approximate solution, and a guaranteed ε -small duality gap. Here the constant $C_f^{\text{prod}} := \sum_{i=1}^n C_f^{(i)}$ is the sum of the (partial) curvature constants of f with respect to the individual domain factors $\mathcal{M}^{(i)}$. We discuss this Lipschitz-assumption on the gradient in more detail in Appendix C, and will compute this constant precisely for the structural SVM in Section 5. In the following convergence results, $h_0 := f(\alpha^{(0)}) - f(\alpha^*)$ is the initial error at the starting point of the algorithm, for $\alpha^* \in \mathcal{M}$ being an optimal solution. Proofs are provided in Appendix E (cf. Theorems 7 and 9).

Theorem 1. *For each $k \geq 0$, the iterate $\alpha^{(k)}$ of Algorithm 2 (either using the predefined step-sizes, or using line-search) satisfies $\mathbb{E}[f(\alpha^{(k)})] - f(\alpha^*) \leq \frac{2n}{k+2n} (2C_f^{\text{prod}} + h_0)$, where $\alpha^* \in \mathcal{M}$ is an optimal solution to problem (6), and the expectation is over the random choice of the factor i in the steps of the algorithm.*

Furthermore, if Algorithm 2 is run for $K \geq 2$ iterations, then it has an iterate $\alpha^{(\hat{k})}$, $1 \leq \hat{k} \leq K$, with expected duality gap bounded by $\mathbb{E}[g(\alpha^{(\hat{k})})] \leq \frac{6n}{K} (2C_f^{\text{prod}} + h_0)$.

4. The Frank-Wolfe Algorithm for Structural SVMs

In this section, we explain how the Frank-Wolfe Algorithm 1 can be efficiently applied to solve the dual problem (4) of the structural SVM and show its relationship to other algorithms. Recall that the optimization domain for the dual variables α is the product of n simplices, $\mathcal{M} = \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$. Since each simplex consists of a potentially exponential number $|\mathcal{Y}_i|$ of dual variables, we cannot maintain a dense vector α during the algorithm. Note though that each “corner” s of this domain corresponds to putting all the mass in each simplex i on a single labeling y_i – i.e. $s := (\mathbf{e}^{y_1}, \dots, \mathbf{e}^{y_n}) \in \mathcal{M}$, where \mathbf{e}^{y_i} is a vector of zeros everywhere except with a one at the coordinate $\alpha_i(y_i)$. The main insight which enables us to apply Frank-Wolfe here is to notice that the linear subproblem used in Frank-Wolfe, which reduces to a search over all corners of the domain, is actually equivalent to solving the loss-augmented decoding subproblem on each datapoint, and thus can be done efficiently (see Appendix D.1 for details). Moreover, as mentioned in section 3, each iterate $\alpha^{(k)}$ of the Frank-Wolfe algorithm is a sparse convex combination of the previously visited corners s and the starting point $\alpha^{(0)}$, and so we would only need to maintain the list of previously seen solutions to the loss-augmented decoding subproblems to keep track of the non-zero coordinates of α , avoiding the problem of its exponential size. Finally, in the case that we do not use kernels, we avoid the quadratic explosion of the number of operations needed in the dual by *not* explicitly maintaining $\alpha^{(k)}$, but rather *only* maintaining explicitly the corresponding *primal* variable vector $w^{(k)} := A\alpha^{(k)}$. The resulting Algorithms 3 and 4 are equivalent to the original optimization Algorithms 1 and 2, but the iterates are only represented in the primal.

Algorithm 3: Batch Primal-Dual Frank-Wolfe Algorithm for the Structural SVM	Algorithm 4: Coordinate Descent Frank-Wolfe Algorithm for the Structural SVM
Let $\mathbf{w}^{(0)} := \mathbf{0}$, $\ell^{(0)} := 0$ for $k = 0 \dots K$ do Let $\gamma := \frac{2}{k+2}$ for $i = 1 \dots n$ do Solve $\mathbf{y}_i := \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}_i} H_i(\mathbf{y}; \mathbf{w}^{(k)})$ end Let $\mathbf{w}_s := \sum_{i=1}^n \frac{1}{\lambda n} \psi_i(\mathbf{y}_i)$ and $\ell_s := \frac{1}{n} \sum_{i=1}^n L_i(\mathbf{y}_i)$ <i>Optionally:</i> Choose γ by line-search Update $\mathbf{w}^{(k+1)} := (1 - \gamma)\mathbf{w}^{(k)} + \gamma \mathbf{w}_s$ and $\ell^{(k+1)} := (1 - \gamma)\ell^{(k)} + \gamma \ell_s$ end	Let $\mathbf{w}^{(0)} := \mathbf{w}_i^{(0)} := \mathbf{0}$, $\ell^{(0)} := \ell_i^{(0)} := 0$ for $k = 0 \dots K$ do Pick $i \in_{u.a.r.} [n]$ Let $\gamma := \frac{2n}{k+2n}$ Solve $\mathbf{y}_i := \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}_i} H_i(\mathbf{y}; \mathbf{w}^{(k)})$ Let $\mathbf{w}_s := \frac{1}{\lambda n} \psi_i(\mathbf{y}_i)$ and $\ell_s := \frac{1}{n} L_i(\mathbf{y}_i)$ <i>Optionally:</i> Choose γ by line-search Update $\mathbf{w}_i^{(k+1)} := (1 - \gamma)\mathbf{w}_i^{(k)} + \gamma \mathbf{w}_s$ and $\ell_i^{(k+1)} := (1 - \gamma)\ell_i^{(k)} + \gamma \ell_s$ Update $\mathbf{w}^{(k+1)} := \mathbf{w}^{(k)} + \mathbf{w}_i^{(k+1)} - \mathbf{w}_i^{(k)}$ and $\ell^{(k+1)} := \ell^{(k)} + \ell_i^{(k+1)} - \ell_i^{(k)}$ end

A Primal-Dual Frank-Wolfe Algorithm for the Structural SVM Dual By applying the Frank-Wolfe Algorithm 1 to the dual of the structural SVM (4), but only maintaining the primal iterates $\mathbf{w}^{(k)} := A\boldsymbol{\alpha}^{(k)}$, we obtain Algorithm 3. Note that the Frank-Wolfe search corner $\mathbf{s} = (\mathbf{e}^{\mathbf{y}_1}, \dots, \mathbf{e}^{\mathbf{y}_n})$, which is obtained by solving the loss-augmented subproblems, yields the update vector $\mathbf{w}_s = A\mathbf{s}$. Furthermore, we have used the natural starting point $\boldsymbol{\alpha}^{(0)} := (\mathbf{e}^{\mathbf{y}_1^D}, \dots, \mathbf{e}^{\mathbf{y}_n^D})$ which yields $\mathbf{w}^{(0)} = \mathbf{0}$ as $\psi_i(\mathbf{y}_i^D) = \mathbf{0}$ for all i . The ℓ quantities are maintained for the computation of the duality gap as well as the line-search step-size (see below).

The Duality Gap The duality gap (5) for our structural SVM dual objective function (4) here is given by $g(\boldsymbol{\alpha}) := \max_{\mathbf{s}' \in \mathcal{M}} \langle \boldsymbol{\alpha} - \mathbf{s}', \nabla f(\boldsymbol{\alpha}) \rangle = (\boldsymbol{\alpha} - \mathbf{s})^T (\lambda A^T A \boldsymbol{\alpha} - \mathbf{b}) = \lambda (\mathbf{w} - A\mathbf{s})^T \mathbf{w} - \mathbf{b}^T \boldsymbol{\alpha} + \mathbf{b}^T \mathbf{s}$, where \mathbf{s} is an *exact* minimizer of the linearized problem given at the point $\boldsymbol{\alpha}$. Note that this (Fenchel) duality gap turns out to be the same as the Lagrangian duality gap (see Appendix B.2), and so gives us a direct handle on the suboptimality error of $\mathbf{w}^{(k)}$ for the primal problem (3). Using $\mathbf{w}_s := A\mathbf{s}$ and $\ell_s := \mathbf{b}^T \mathbf{s}$, we observe that the gap is well-defined and efficient to compute when we only have the *primal* variables $\mathbf{w} = A\boldsymbol{\alpha}$ and $\ell = \mathbf{b}^T \boldsymbol{\alpha}$ available, and becomes

$$g(\mathbf{w}, \ell, \mathbf{w}_s, \ell_s) := \lambda (\mathbf{w} - \mathbf{w}_s)^T \mathbf{w} - \ell + \ell_s = g(\boldsymbol{\alpha}). \quad (7)$$

Since the quantities $\mathbf{w}, \ell, \mathbf{w}_s, \ell_s$ are maintained during the run of Algorithm 3, we can keep track of the duality gap, and use $g(\boldsymbol{\alpha}^{(k)}) = g(\mathbf{w}^{(k)}, \ell^{(k)}, \mathbf{w}_s, \ell_s) \leq \varepsilon$ as the proper stopping criterion.

Implementing the Line-Search Because the objective of the structural SVM dual (4) is a quadratic function in $\boldsymbol{\alpha}$, the optimal step-size for any given candidate search point $\mathbf{s} \in \mathcal{M}$ can be obtained analytically with a simple formula. Namely, if we let $\gamma_{LS} = \operatorname{argmin}_{\gamma \in [0,1]} f(\boldsymbol{\alpha} + \gamma(\mathbf{s} - \boldsymbol{\alpha}))$, we have that $\gamma_{LS} := \max\{0, \min\{1, \gamma_{opt}\}\}$, where γ_{opt} is obtained by setting the derivative of the corresponding univariate quadratic function in γ to zero, which here gives $\gamma_{opt} := \frac{\langle \boldsymbol{\alpha} - \mathbf{s}, \nabla f(\boldsymbol{\alpha}) \rangle}{\lambda \|A(\boldsymbol{\alpha} - \mathbf{s})\|^2} = \frac{g(\mathbf{w}, \ell, \mathbf{w}_s, \ell_s)}{\lambda \|\mathbf{w} - \mathbf{w}_s\|^2}$. The first equality is valid for *any* search point $\mathbf{s} \in \mathcal{M}$ (and so can also be used for the other variant described in Algorithm 4 or for an approximate search point), whereas the second equality (with the duality gap) is only valid for \mathbf{s} being the exact minimizer of the linearized problem at $\boldsymbol{\alpha}$. In both cases, all the necessary quantities to compute γ_{LS} are maintained during the run of Algorithm 3, as was the case for the duality gap.

Convergence Proof and Running Time In the following, we write R for the maximal length of a difference feature vector, i.e. $R := \max_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \|\boldsymbol{\psi}_i(\mathbf{y})\|_2$, and that the maximum error $L_{\max} := \max_{i, \mathbf{y}} L_i(\mathbf{y})$. By bounding the curvature constant C_f for the objective function f being the dual SVM objective (4), we can now directly apply the known convergence results for the standard Frank-Wolfe algorithm. This then directly leads to the main result of this section, that the number of iterations of Algorithm 3 is independent of the data size n .

Theorem 2 (Batch FW SVM). *Algorithm 3 obtains an ε -approximate solution to the structural SVM dual problem (4) and duality gap $g(\boldsymbol{\alpha}^{(k)}) \leq \varepsilon$ after at most $O\left(\frac{R^2}{\lambda \varepsilon}\right)$ iterations, where each iteration costs n oracle calls.*

Since we have proved that the duality gap is smaller than ε , this implies that the original SVM primal objective (3) is actually solved to accuracy ε as well.

Relationship with Batch Subgradient Descent in the Primal The batch Frank-Wolfe Algorithm 3 is equivalent to subgradient descent in the primal, though with a clever choice of step-size in case the line-search in the dual is used. To see this, notice that a subgradient of (3) is given by $\mathbf{d}_{\text{sub}} = \lambda \mathbf{w} - \frac{1}{n} \sum_i \boldsymbol{\psi}_i(\mathbf{y}_i) = \lambda(\mathbf{w} - \mathbf{w}_s)$, where \mathbf{y}_i and \mathbf{w}_s are as defined in Algorithm 3. Hence, for a step-size of β , the subgradient descent update becomes $\mathbf{w}^{(k+1)} := \mathbf{w}^{(k)} - \beta \mathbf{d}_{\text{sub}} = \mathbf{w}^{(k)} - \beta \lambda (\mathbf{w}^{(k)} - \mathbf{w}_s) = (1 - \beta \lambda) \mathbf{w}^{(k)} + \beta \lambda \mathbf{w}_s$. Comparing this with Algorithm 3, we see that each Frank-Wolfe step on the dual problem (4) with step-size γ is equivalent to a batch subgradient descent step in the primal with a step-size of $\beta = \gamma/\lambda$, and thus our convergence results also apply to it. This seems to generalize the equivalence between Frank-Wolfe optimization and subgradient descent for a quadratic objective with identity Hessian which was already observed in Bach et al. (2012, Section 4.1) and in Bach (2011, Section 6.3).

Relationship with Cutting Plane Algorithms The cutting plane algorithm of (Joachims et al., 2009) (either in its 1-slack or n -slack version) finds iteratively some new coordinates to add to the dual problem (which equivalently correspond to constraints in the primal problem) by solving the same loss-augmented decoding problem for each datapoint that we use in the batch Frank-Wolfe algorithm. But instead of doing a line search towards the corner \mathbf{s} , as is done in Frank-Wolfe, it re-optimizes the QP over all the previously added constraints. The minimum-norm-point extension of Frank-Wolfe, which in each iteration re-optimizes over all previously visited coordinates (Clarkson, 2010), can thus be seen equivalent to it. The minimum-norm-point convergence results simply reuse the one from Frank-Wolfe with line search, thus all our results apply as well to the cutting plane algorithm of (Joachims et al., 2009).

5. Faster Frank-Wolfe Coordinate Descent for the Structural SVM Dual

Algorithm 4 represents our new coordinate descent Frank-Wolfe Algorithm 2 applied to the SVM-dual problem (4). Using that the updates \mathbf{w}_s and ℓ_s correspond to $\mathbf{w}_s = \mathbf{A} \mathbf{s}_{[i]}$ and $\ell_s = \mathbf{b}^T \mathbf{s}_{[i]}$ where $\mathbf{s}_{[i]}$ is the padding with zeros of $\mathbf{s}_{(i)} := \mathbf{e}^{\mathbf{y}_i} \in \mathcal{M}^{(i)}$ so that $\mathbf{s}_{[i]} \in \mathcal{M}$, we obtain that Algorithm 4 is actually equivalent to Algorithm 2.

Convergence Proof and Running Time In order to apply the convergence result of our new product Frank-Wolfe algorithm to the SVM case, we need to bound the total curvature constant C_f^{prod} for the dual SVM problem. This then directly leads to the following theorem, showing that the total running time of Algorithm 4 is independent of the data size n , when measured in the number of oracle calls. In other words, the theory says that our product Algorithm 4

is n times faster when compared to the batch Frank-Wolfe version in Algorithm 3, where each iteration requires n oracle calls. This speed-up is enabled by the fact that for the SVM dual optimization problem, the product curvature C_f^{prod} turns out to be n times smaller than the classical Frank-Wolfe curvature constant C_f . We will also practically observe this running time difference in the experiments in Section 7.

Theorem 3 (Stochastic FW SVM). *If $L_{\max} \leq \frac{2R^2}{\lambda n}$, then Algorithm 4 obtains an ε -approximate solution to the structural SVM dual problem (4) and expected duality gap $\mathbb{E}[g(\boldsymbol{\alpha}^{(k)})] \leq \varepsilon$ after at most $O\left(\frac{R^2}{\lambda \varepsilon}\right)$ iterations, where each iteration costs a single oracle call.*

If $L_{\max} > \frac{2R^2}{\lambda n}$, then the line-search variant of Algorithm 4 will require an additional (constant in ε) number of $2n \log\left(\frac{\lambda n L_{\max}}{2R^2}\right)$ steps to get the same error and duality gap guarantees, whereas the predefined step-size variant will require an additional $O\left(\frac{n L_{\max}}{\varepsilon}\right)$ steps.

6. Extensions

Approximate Linear Subproblems and Approximate Decoding Interestingly, it can be shown that the convergence results we presented above also hold if approximate minimizers of the linear subproblems are used instead of exact minimizers. More formally, we require that the step direction $\mathbf{s}_{(i)}$ in Algorithm 2 (or \mathbf{s} in Algorithm 1) is chosen such that $\langle \mathbf{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha}^{(k)}) \rangle \leq \text{argmin}_{\mathbf{s}_{(i)} \in \mathcal{M}^{(i)}} \langle \mathbf{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha}^{(k)}) \rangle + \varepsilon_k$, where the additive approximation quality is defined as $\varepsilon_k := \gamma_k C_f^{(i)}$, for γ_k being the predefined step size. With a step-oracle of this accuracy, the above convergence bounds from Theorem 1 do still apply to the approximate version of Algorithm 2 (and so Algorithm 1 using $n = 1$). The only change being that the upper bound on the errors are multiplied by a factor of two. A proof of this generalization is also provided in Appendix E.

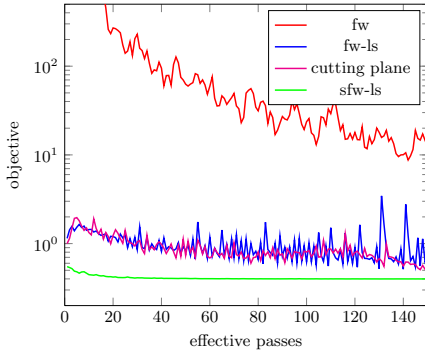
This makes the algorithm more applicable to large-scale applications where it is too costly to do the linear optimization exactly. In the case of structural SVMs, this means that we can run the above mentioned algorithms with approximate loss-augmented decoders which is crucial for many applications.

Kernelized Algorithm Both Algorithms 3 and 4 can be used with kernels by maintaining explicitly the sparse dual variables $\boldsymbol{\alpha}^{(k)}$ instead of the primal variables $\mathbf{w}^{(k)}$. In this case, the classifier is only given implicitly as a sparse combination of the corresponding kernel functions, i.e. $\mathbf{w} = A\boldsymbol{\alpha}$. Using our Algorithm 4, we obtain the currently best known bound on the number of support vectors, i.e. a guaranteed ε -approximation consisting of only $\frac{R^2}{\lambda \varepsilon}$ many support vectors. For comparison, the standard cutting plane method (Joachims et al., 2009) adds n support vectors $\boldsymbol{\psi}_i(\mathbf{y})$ at each of their iterations. A more detailed version of the kernelized variant of Algorithm 4 is given in Appendix D.4.

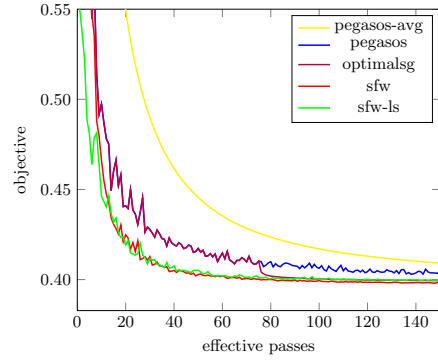
7. Experiments

Here we compare existing algorithms for solving the structural SVM problem to our novel Frank-Wolfe approaches. The different methods are applied to the OCR dataset from (Taskar et al., 2003) and the CoNLL dataset (Sang and Buchholz, 2000). Both datasets correspond to sequence labeling tasks, for which the loss-augmented decoding problem is solved exactly by the Viterbi algorithm. The Frank-Wolfe variants studied are: batch Frank-Wolfe with and without line-search (fw-ls and fw) as in Algorithm 3 and stochastic Frank-Wolfe with and without line-search (sfw-ls and sfw), see Algorithm 4. We include the following competing methods in the comparison: The cutting plane algorithm implemented in SVMstruct (Joachims et al., 2009) with its default options, standard and averaged Pegasos (Shalev-Shwartz et al., 2010) and the optimal stochastic subgradient method from (Rakhlin et al., 2012) which is the same as Pegasos but the averaging is

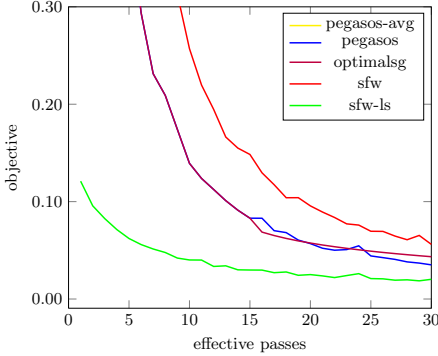
only started after half of the iterations (which yields a convergence rate of $O(1/k)$ vs. $O(\log k/k)$ for Pegasos). The progress of the different algorithms is visualized in Figure 1. Figure 1a compares the batch algorithms to the stochastic Frank-Wolfe algorithm with line-search. We observe that in about 20 passes through the dataset, stochastic FW achieves a solution close to the optimum, whereas the batch solvers do not converge even within 150 iterations. The improved convergence of the stochastic algorithms when compared to the batch solvers was systematic in all experiments for large λ . However, for small values of λ , the stochastic approaches without line-search, such as Pegasos, often perform worse than the batch algorithms. The suggested stochastic version of Frank-Wolfe with line-search does not share this weakness and performs well in both settings. Figure 1b and 1c show the progression of the stochastic solvers for the OCR and the CoNLL datasets. In both of these experiments Algorithm 4 with line-search outperforms the other stochastic algorithms. The improvement is especially large in the first iterations. We conjecture this is due to the line-search advantage, as there is also an improvement in the first few passes for sfw-ls vs. sfw. Finally, Figure 1d shows the test error during the optimization – there, the stochastic Frank-Wolfe algorithm still shows an advantage, though smaller than for the primal objective.



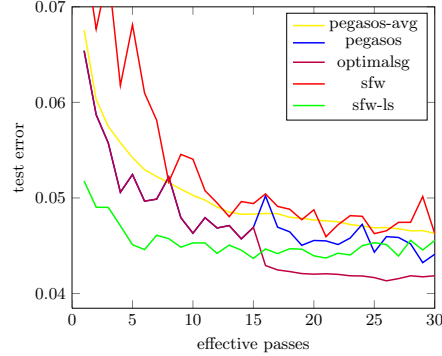
(a) Batch vs. stochastic for $\lambda = 0.01$ on OCR.



(b) Stochastic solvers for $\lambda = 0.01$ on OCR.



(c) Primal objective for $\lambda = 0.001$ on CoNLL.



(d) Test error for $\lambda = 0.001$ on CoNLL.

Figure 1: In our experiments the stochastic Frank-Wolfe algorithm (sfw-ls) with line-search dominates all competing approaches. See text for details.

8. Related Work

There has been a fair amount of work for coordinate descent methods for the dual of binary SVM such as with the original SMO algorithm. The SMO algorithm was generalized for the factored representation of the Max-Margin Markov network version of structural SVMs (and thus using something equivalent to an expectation oracle) in Taskar (2004, Chapter 6), but its convergence

rate scales badly with the size of the output space: it was estimated as $O(n|\mathcal{Y}|/\lambda\epsilon)$ in Zhang et al. (2011). This is the only coordinate descent method for the dual of structural SVM with rate guarantees that we are aware of (in addition to the new stochastic Frank-Wolfe algorithm). Rousu proposed in Rousu et al. (2006) another coordinate descent method on the factored dual: they pick one sample at a time and optimize on the relevant subspace using multiple Frank-Wolfe updates, though with no global rate guarantees. Interestingly, our stochastic Frank-Wolfe algorithm applied to binary SVM reduces to the dual descent method proposed in Hsieh et al. (2008), which samples one random datapoint at a time to update its associated unique dual variable and so exact coordinate optimization can be accomplished (in our formulation, we get a 2-simplex for binary SVM and so the line search also gives the exact subspace optimizer). In this case, they prove a locally linear rate of convergence, thanks to the exact subspace optimization. We note that our duality gap guarantee for stochastic Frank-Wolfe thus clarifies that the dual descent method of (Hsieh et al., 2008) also yields a primal convergence guarantee of $O(1/k)$, thus improving over Pegasos – this seems to have been observed empirically in Shalev-Shwartz et al. (2010). The work of (Hsieh et al., 2008) was generalized to the structural SVM in Balamurugan et al. (2011) in a sequential dual minimization approach where a QP on each example is approximately solved sequentially using SMO, but with no rate guarantees. Finally, we note that few guarantees are given for the optimization of structural SVMs with approximate oracles. The cutting plane algorithm of (Tsochantaridis et al., 2005) was analyzed in Finley and Joachims (2008) when the maximization oracle gives a *multiplicative* error, though the dependence of the running time of this algorithm to achieve an ϵ -approximate solution in term of this multiplicative error was left unclear. In contrast, we provide guarantees for batch subgradient, cutting plane as well as the stochastic Frank-Wolfe algorithm to achieve an ϵ -approximate solution as long as the *additive* error of the oracle is within ϵ . We get stronger guarantees, but also with a stronger assumption, as the approximation errors of maximization oracles are usually specified with a multiplicative factor.

9. Conclusion

We highlighted the equivalence of batch Frank-Wolfe on the dual of structural SVM with batch subgradient in the primal, thereby obtaining a line search version which had better robustness and improving on its rate analysis. We conjecture that this kind of equivalence could be generalized to other quadratic objectives and thus could provide additional insights on other traditional algorithms. We proposed a new stochastic coordinate descent Frank-Wolfe algorithm on arbitrary compact product domains with provable convergence guarantees. When applying it to structural SVMs, we obtain a simple online algorithm which converges empirically faster than other stochastic algorithms on the first few passes of the data, thanks to an increased robustness from the line search. As step-size selection is a notoriously hard problem for stochastic subgradient methods, the line search gives it a significant advantage. The stochastic coordinate descent Frank-Wolfe optimization algorithm being applicable to general compact product domains and convex functions, we believe that it can provide a new optimization tool of choice in the machine learning toolbox.

Acknowledgements We thank Francis Bach, Bernd Gärtner and Ronny Luss for helpful discussions, and Robert Carnecky for the 3D illustration of Frank-Wolfe. Martin Jaggi is supported by the ERC Project SIPA, and by the Swiss National Science Foundation (SNSF). Simon Lacoste-Julien and Mark Schmidt are partly supported by the European Research Council (SIERRA-ERC-239993). Simon Lacoste-Julien is supported by a Research in Paris fellowship and Mark Schmidt is supported by a postdoctoral fellowship from the National Science and Engineering Research Council of Canada.

References

- F. Bach. Learning with submodular functions: a convex optimization perspective. Nov. 2011.
- F. Bach, S. Lacoste-Julien, and G. Obozinski. On the equivalence between herding and conditional gradient algorithms. In *ICML*, 2012.
- P. Balamurugan, S. Shevade, S. Sundararajan, and S. Keerthi. A sequential dual method for structural SVMs. In *SDM*, 2011.
- D. Bertsekas. *Nonlinear programming*. Athena Scientific, Belmont, MA, 1999.
- J. Borwein and A. Lewis. *Convex analysis and nonlinear optimization: theory and examples*. 2006.
- S. Boyd and L. Vandenberghe. *Convex optimization*. 2004.
- T. Caetano, J. McAuley, L. Cheng, Q. Le, and A. Smola. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1048–1058, 2009.
- K. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms*, 6(4):1–30, 2010.
- M. Collins, A. Globerson, T. Koo, X. Carreras, and P. L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *JMLR*, 9:1775–1822, 2008.
- J. Dunn and S. Harshbarger. Conditional gradient algorithms with open loop step size rules. *Journal of Mathematical Analysis and Applications*, 62(2):432–444, 1978.
- T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *ICML*, 2008.
- M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3: 95–110, 1956.
- B. Gärtner and M. Jaggi. Coresets for polytope distance. *ACM SCG*, 2009.
- C. Hsieh, K. Chang, C. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML*, pages 408–415, 2008.
- M. Jaggi. *Sparse convex optimization methods for machine learning*. PhD thesis, ETH Zürich, 2011.
- T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1): 27–59, 2009.
- S. Keerthi, S. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, 2000.
- O. Mangasarian. Machine learning via polyhedral concave minimization. Technical report, 1995.
- Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems, 2010.
- H. Ouyang and A. Gray. Fast stochastic Frank-Wolfe algorithms for nonlinear SVMs. *SDM*, 2010.
- M. Patriksson. Decomposition methods for differentiable optimization problems over cartesian product sets. *Computational Optimization and Applications*, 9(1):5–42, 1998.
- A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, 2012.
- N. Ratliff, J. A. Bagnell, and M. Zinkevich. (online) subgradient methods for structured prediction. In *AISTATS*, 2007.
- P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *arXiv.org*, math.OC, July 2011.

- J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *JMLR*, 2006.
- E. Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking, 2000.
- S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1), 2010.
- B. Taskar. *Learning structured prediction models: A large margin approach*. PhD thesis, Stanford, 2004.
- B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2003.
- B. Taskar, S. Lacoste-Julien, and M. I. Jordan. Structured prediction, dual extragradient and Bregman projections. *JMLR*, 7:1627–1653, 2006.
- C. Teo, A. Smola, S. Vishwanathan, and Q. Le. A scalable modular convex solver for regularized risk minimization. *ACM SIGKDD*, pages 727–736, 2007.
- C. Teo, S. Vishwanathan, A. Smola, and Q. Le. Bundle methods for regularized risk minimization. *JMLR*, 11:311–365, 2010.
- I. Tschantz, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.
- X. Zhang, A. Saha, and S. V. N. Vishwanathan. Accelerated training of max-margin Markov networks with kernels. In *ALT*, pages 292–307. Springer-Verlag, 2011.

A. Equivalence of the “Linearization”-Duality Gap to a Special Case of Fenchel Duality

For our used constrained optimization framework, the notion of the simple duality gap was crucial. Consider a general constrained optimization problem

$$\min_{\mathbf{x} \in \mathcal{M}} f(\mathbf{x}) , \quad (8)$$

where the domain (or feasible set) $\mathcal{M} \subseteq \mathcal{X}$ is an arbitrary compact subset of a Euclidean space \mathcal{X} . We assume that the objective function f is convex, but not necessarily differentiable.

In this case, the general “linearization” duality gap (5) as proposed by (Jaggi, 2011, Section 2.2) is given by

$$g(\mathbf{x}; d_{\mathbf{x}}) = \mathbf{I}_{\mathcal{M}}^*(-d_{\mathbf{x}}) + \langle \mathbf{x}, d_{\mathbf{x}} \rangle . \quad (9)$$

Here $d_{\mathbf{x}}$ is an arbitrary subgradient to f at the candidate position \mathbf{x} , and $\mathbf{I}_{\mathcal{M}}^*(\mathbf{y}) := \sup_{\mathbf{s} \in \mathcal{M}} \langle \mathbf{s}, \mathbf{y} \rangle$ is the *support function* of the set \mathcal{M} .

Convexity of f implies that the linearization $f(\mathbf{x}) + \langle \mathbf{s} - \mathbf{x}, d_{\mathbf{x}} \rangle$ always lies below the graph of the function f , as illustrated by the figure in Section 3. This immediately gives the crucial property of the duality gap (9), as being a *certificate* for the current approximation quality, i.e. upper-bounding the (unknown) error $g(\mathbf{x}) \geq f(\mathbf{x}) - f(\mathbf{x}^*)$, where \mathbf{x}^* is some optimal solution.

Note that for differentiable functions f , the gradient is the unique subgradient at \mathbf{x} , therefore the duality gap equals $g(\mathbf{x}) := g(\mathbf{x}; \nabla f(\mathbf{x}))$ as we defined in (5).

Fenchel Duality Here we will additionally explain how the duality gap (9) can also be interpreted as a special case of standard Fenchel convex duality.

We consider the equivalent formulation of our constrained problem (8), given by

$$\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) + \mathbf{I}_{\mathcal{M}}(\mathbf{x}) .$$

Here the *set indicator function* $\mathbf{I}_{\mathcal{M}}$ of a subset $\mathcal{M} \subseteq \mathcal{X}$ is defined as $\mathbf{I}_{\mathcal{M}}(\mathbf{x}) := 0$ for $\mathbf{x} \in \mathcal{M}$ and $\mathbf{I}_{\mathcal{M}}(\mathbf{x}) := +\infty$ for $\mathbf{x} \notin \mathcal{M}$.

The *Fenchel conjugate* function f^* of a function f is given by $f^*(\mathbf{y}) := \sup_{\mathbf{x} \in \mathcal{X}} \langle \mathbf{x}, \mathbf{y} \rangle - f(\mathbf{x})$.

For example, observe that the Fenchel conjugate of a set indicator function $\mathbf{I}_{\mathcal{M}}(\cdot)$ is given by its support function $\mathbf{I}_{\mathcal{M}}^*(\cdot)$.

From the above definition of the conjugate, the *Fenchel-Young inequality* $f(\mathbf{x}) + f^*(\mathbf{y}) \geq \langle \mathbf{x}, \mathbf{y} \rangle \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$ follows directly.

Now we consider the *Fenchel dual problem* of minimizing $p(\mathbf{x}) := f(\mathbf{x}) + \mathbf{I}_{\mathcal{M}}(\mathbf{x})$, which is defined as to maximize $d(\mathbf{y}) := -f^*(\mathbf{y}) - \mathbf{I}_{\mathcal{M}}^*(-\mathbf{y})$. By the Fenchel-Young inequality, and assumed that $\mathbf{x} \in \mathcal{M}$, we have that $\forall \mathbf{y} \in \mathcal{X}$,

$$\begin{aligned} p(\mathbf{x}) - d(\mathbf{y}) &= f(\mathbf{x}) - (-f^*(\mathbf{y}) - \mathbf{I}_{\mathcal{M}}^*(-\mathbf{y})) \\ &\geq \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{I}_{\mathcal{M}}^*(-\mathbf{y}) \\ &= g(\mathbf{x}; \mathbf{y}) . \end{aligned}$$

Furthermore, this inequality becomes an equality if and only if \mathbf{y} is chosen as a subgradient to f at \mathbf{x} , that is if $\mathbf{y} := -d_{\mathbf{x}}$. The last fact follows from the known equivalent characterization of the subdifferential in terms of the Fenchel conjugate: $\partial f(\mathbf{x}) := \{\mathbf{y} \in \mathcal{X} \mid f(\mathbf{x}) + f^*(\mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle\}$. For a more detailed explanation of Fenchel duality, we refer the reader to the standard literature, e.g. Borwein and Lewis (2006, Theorem 3.3.5).

To summarize, we have obtained that the simpler “linearization” duality gap $g(\mathbf{x}; d_{\mathbf{x}})$ as given in (9) is indeed the difference of the current objective to the Fenchel dual problem, when being restricted to the particular choice of the dual variable \mathbf{y} being a subgradient at the current position \mathbf{x} .

B. Derivations of the SVM Dual Formulations

B.1. Derivation of the n-Slack Dual

Proof of the dual of the n-Slack-Formulation. See also Collins et al. (2008). For a self-contained explanation of Lagrange duality we refer the reader to Boyd and Vandenberghe (2004, Section 5). The Lagrangian

of (1) is

$$L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{\lambda}{2} \langle \mathbf{w}, \mathbf{w} \rangle + \frac{1}{n} \sum_{i=1}^n \xi_i + \sum_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \frac{1}{n} \alpha_i(\mathbf{y}) (-\xi_i + \langle \mathbf{w}, -\boldsymbol{\psi}_i(\mathbf{y}) \rangle + L_i(\mathbf{y})),$$

where $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_n) \in \mathbb{R}^{|\mathcal{Y}_1|} \times \dots \times \mathbb{R}^{|\mathcal{Y}_n|} = \mathbb{R}^m$ are the corresponding (non-negative) Lagrange multipliers. Here we have re-scaled the multipliers (dual variables) by a constant of $\frac{1}{n}$, corresponding to multiplying the corresponding original primal constraint by $\frac{1}{n}$ on both sides, which does not change the optimization problem.

Since the objective as well as the constraints are continuously differentiable with respect to $(\mathbf{w}, \boldsymbol{\xi})$, the Lagrangian L will attain its finite minimum over $\boldsymbol{\alpha}$ when $\nabla_{(\mathbf{w}, \boldsymbol{\xi})} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0$. Making this saddle-point condition explicit results in a simplified Lagrange dual problem, which is also known as the Wolfe dual. In our case, this condition from differentiating w.r.t. \mathbf{w} is

$$\lambda \mathbf{w} = \sum_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \frac{1}{n} \alpha_i(\mathbf{y}) \boldsymbol{\psi}_i(\mathbf{y}). \quad (10)$$

And differentiating with respect to ξ_i and setting the derivatives to zero gives²

$$\sum_{\mathbf{y} \in \mathcal{Y}_i} \alpha_i(\mathbf{y}) = 1 \quad \forall i \in [n].$$

Plugging this condition and the expression (10) for \mathbf{w} back into the Lagrangian, we obtain the Lagrange dual problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{\lambda}{2} \left\| \sum_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \alpha_i(\mathbf{y}) \frac{\boldsymbol{\psi}_i(\mathbf{y})}{\lambda n} \right\|^2 + \sum_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \alpha_i(\mathbf{y}) \frac{L_i(\mathbf{y})}{n} \\ \text{s.t.} \quad & \sum_{\mathbf{y} \in \mathcal{Y}_i} \alpha_i(\mathbf{y}) = 1 \quad \forall i \in [n], \\ & \text{and } \alpha_i(\mathbf{y}) \geq 0 \quad \forall i \in [n], \forall \mathbf{y} \in \mathcal{Y}_i, \end{aligned}$$

which is exactly the negative of the quadratic program claimed in (4). \square

B.2. Relation between the Fenchel Gap and the Classical Lagrange Duality Gap

Consider the difference of our objective at $\mathbf{w} := A\boldsymbol{\alpha}$ in the primal problem (1), and the dual objective at $\boldsymbol{\alpha}$ in problem (4) (in the maximization version). This difference is

$$\begin{aligned} g_{\text{Lagrange}}(\mathbf{w}, \boldsymbol{\alpha}) &= \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} + \frac{1}{n} \sum_{i=1}^n \xi_i - \left(\mathbf{b}^T \boldsymbol{\alpha} - \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \right) \\ &= \lambda \mathbf{w}^T \mathbf{w} - \mathbf{b}^T \boldsymbol{\alpha} + \frac{1}{n} \sum_{i=1}^n \xi_i \end{aligned}$$

We now suppose that we use the minimizing slack variables for the primal problem (1) (i.e. we are computing the primal objective (3)) – thus we set $\xi_i = \max_{\mathbf{y} \in \mathcal{Y}_i} H_i(\mathbf{y}; \mathbf{w})$ (cf. (2)). Furthermore, we use that by the definition of A, \mathbf{b} , we have $H_i(\mathbf{y}; \mathbf{w}) = n(\mathbf{b} - \lambda A^T \mathbf{w})_{(i, \mathbf{y})}$. Summing up over all points, we obtain

$$\frac{1}{n} \sum_{i=1}^n \xi_i = \frac{1}{n} \sum_{i=1}^n \max_{\mathbf{y} \in \mathcal{Y}_i} H_i(\mathbf{y}; \mathbf{w}) = \sum_{i=1}^n \max_{\mathbf{y} \in \mathcal{Y}_i} (\mathbf{b} - \lambda A^T \mathbf{w})_{(i, \mathbf{y})}.$$

Now the equivalence of the two expressions follows along the same lines as in Lemma 6 on the definition of the update direction \mathbf{s} . Formally, since \mathbf{s} in Algorithm 3 was precisely defined to give $\frac{1}{n} \sum_{i=1}^n \xi_i = \sum_{i=1}^n \max_{\mathbf{y} \in \mathcal{Y}_i} (\mathbf{b} - \lambda A^T \mathbf{w})_{(i, \mathbf{y})} = \mathbf{s}^T (\mathbf{b} - \lambda A^T \mathbf{w})$, we conclude that the standard Lagrangian gap is indeed identical to the used duality gap as defined in (5) and (7), i.e.

$$g_{\text{Lagrange}}(\mathbf{w}, \boldsymbol{\alpha}) = \lambda (\mathbf{w} - A\mathbf{s})^T \mathbf{w} - \mathbf{b}^T \boldsymbol{\alpha} + \mathbf{b}^T \mathbf{s}.$$

²Note that because the Lagrangian is linear in ξ_i , if this condition is not satisfied, the minimization of the Lagrangian in ξ_i yield $-\infty$ and so these points can be excluded.

C. The Curvature Constants C_f and C_f^{prod}

The Curvature Constant C_f The *curvature constant* C_f (alternatively also called the *strong smoothness* constant of f), is given by the relative deviation of our function f from its linear approximations, over the domain \mathcal{M} . Formally,

$$C_f := \sup_{\substack{x, s \in \mathcal{M}, \\ \gamma \in [0, 1], \\ y = x + \gamma(s - x)}} \frac{1}{\gamma^2} (f(y) - f(x) - \langle y - x, \nabla f(x) \rangle) . \quad (11)$$

It is known that C_f is upper bounded by the Lipschitz-constant of the gradient ∇f times the squared diameter of \mathcal{M} (Clarkson, 2010; Jaggi, 2011).

The Product Curvature Constant C_f^{prod} The curvature concept can be generalized to our setting of product domains $\mathcal{M} := \mathcal{M}^{(1)} \times \dots \times \mathcal{M}^{(n)}$ as follows: over each individual factor, the curvature is given by

$$C_f^{(i)} := \sup_{\substack{x \in \mathcal{M}, s_{(i)} \in \mathcal{M}^{(i)}, \\ \gamma \in [0, 1], \\ y = x + \gamma(s_{(i)} - x_{(i)})}} \frac{1}{\gamma^2} (f(y) - f(x) - \langle y_{(i)} - x_{(i)}, \nabla_{(i)} f(x) \rangle) , \quad (12)$$

where $x_{[i]}$ is $x_{(i)}$ padded with zeros so that $x_{[i]} \in \mathcal{M}$. By considering the Taylor expansion of f , it is not hard to see that also the “partial” curvature $C_f^{(i)}$ is upper bounded by the Lipschitz-constant of the partial gradient $\nabla_{(i)} f$ times the squared diameter of just one domain factor $\mathcal{M}^{(i)}$. See also in the proof of Lemma 5 below.

We define the global *product curvature constant* as the sum of these curvatures for each factor, i.e.

$$C_f^{\text{prod}} := \sum_{i=1}^n C_f^{(i)} \quad (13)$$

Observe that for the classical Frank-Wolfe case when $n = 1$, we recover the original duality gap as well as curvature constant.

Computing the Curvature Constant C_f in the SVM case

Lemma 4. For the dual structural SVM objective function (4) over the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$, the curvature constant C_f , as defined in (11), is upper bounded by

$$C_f \leq \frac{R^2}{\lambda} ,$$

where R is the maximal length of a difference feature vector, i.e. $R := \max_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \|\psi_i(\mathbf{y})\|_2$.

Proof of Lemma 4. If the function is twice differentiable, we can plug-in the second degree Taylor expansion of f into the above definition (11) of the curvature, see e.g. Jaggi (2011, Inequality (2.12)) or Clarkson (2010, Section 4.1). In our case, the gradient at α is given by $\lambda A^T A \alpha - \mathbf{b}$, so that the Hessian is $\lambda A^T A$, being a constant matrix independent of α . This gives the following upper bound on C_f , which we can separate into two identical matrix-vector products with our matrix A :

$$\begin{aligned} C_f &\leq \sup_{\substack{x, y \in \mathcal{M}, \\ z \in [x, y] \subseteq \mathcal{M}}} \frac{1}{2} (y - x)^T \nabla^2 f(z) (y - x) \\ &= \frac{\lambda}{2} \cdot \sup_{x, y \in \mathcal{M}} (A(y - x))^T A(y - x) \\ &= \frac{\lambda}{2} \cdot \sup_{v, w \in A\mathcal{M}} \|w - v\|_2^2 \leq \lambda \cdot \sup_{v \in A\mathcal{M}} \|v\|_2^2 \end{aligned}$$

By definition of our compact domain \mathcal{M} , we have that each vector $v \in A\mathcal{M}$ is precisely the sum of n vectors, each of these being a convex combination of the feature vectors for the possible labelings for data point i .

Therefore, the norm $\|v\|_2$ is upper bounded by n times the longest column of the matrix A , or more formally $\|v\|_2 \leq n \frac{1}{\lambda n} R$ with R being the longest³ feature vector, i.e.

$$R := \max_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \|\psi_i(\mathbf{y})\|_2.$$

Altogether, we have obtained that the curvature C_f is upper bounded by $\frac{R^2}{\lambda}$. \square

Computing the Product Curvature Constant C_f^{prod} in the SVM case

Lemma 5. *For the dual structural SVM objective function (4) over the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$, the total curvature constant C_f^{prod} on the product domain \mathcal{M} , as defined in (13), is upper bounded by*

$$C_f^{\text{prod}} \leq \frac{R^2}{\lambda n}$$

where R is the maximal length of a difference feature vector, i.e. $R := \max_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \|\psi_i(\mathbf{y})\|_2$.

Proof. We follow the same lines as in the above proof, but now applying the same bound to the definition (12) of the curvature on the i -th factor. Here, the change from x to y is now restricted to only affect the coordinates in the i -th factor $\mathcal{M}^{(i)}$. To simplify the notation, let $\mathcal{M}^{[i]}$ be $\mathcal{M}^{(i)}$ augmented with the zero domain for all the other blocks – i.e. the analog of $x_{(i)} \in \mathcal{M}^{(i)}$ is $x_{[i]} \in \mathcal{M}^{[i]}$. $x_{(i)}$ is the i -th block of x whereas $x_{[i]} \in \mathcal{M}$ is $x_{(i)}$ padded with zeros for all the other blocks. We thus require that $y - x \in \mathcal{M}^{[i]}$ for a valid change from x to y . Again by the degree-two Taylor expansion, we obtain

$$\begin{aligned} C_f^{(i)} &\leq \sup_{\substack{x, y \in \mathcal{M}, \\ (y-x) \in \mathcal{M}^{[i]}, \\ z \in [x, y] \subseteq \mathcal{M}}} \frac{1}{2} (y-x)^T \nabla^2 f(z) (y-x) \\ &= \frac{\lambda}{2} \cdot \sup_{\substack{x, y \in \mathcal{M} \\ (y-x) \in \mathcal{M}^{[i]}}} (A(y-x))^T A(y-x) \\ &= \frac{\lambda}{2} \cdot \sup_{v, w \in A\mathcal{M}^{(i)}} \|w - v\|_2^2 \leq \lambda \cdot \sup_{v \in A\mathcal{M}^{(i)}} \|v\|_2^2 \end{aligned}$$

In other words, by definition of our compact domain $\mathcal{M}^{(i)} = \Delta_{|\mathcal{Y}_i|}$, we have that each vector $v \in A\mathcal{M}^{(i)}$ is a convex combination of the feature vectors corresponding to the possible labelings for data point i . Therefore, the norm $\|v\|_2$ is again upper bounded by the longest column of the matrix A , which means $\|v\|_2 \leq \frac{1}{\lambda n} R$ with $R := \max_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \|\psi_i(\mathbf{y})\|_2$. Summing up over the n many factors $\mathcal{M}^{(i)}$, we obtain that the product curvature C_f^{prod} is upper bounded by $\frac{R^2}{\lambda n}$. \square

D. More Details on the Algorithms for Structural SVMs

D.1. Equivalence of an Exact Frank-Wolfe Step and Loss-Augmented Decoding

To see that the proposed Algorithm 3 indeed exactly corresponds to the standard Frank-Wolfe Algorithm 1 applied to the SVM dual problem (4), we verify that the search direction \mathbf{s} giving the update $\mathbf{w}_s = A\mathbf{s}$ is in fact an exact Frank-Wolfe step, which can be seen as follows:

Lemma 6. *The sparse vector $\mathbf{s} \in \mathbb{R}^n$ constructed in the inner for-loop of Algorithm 3 is an exact solution to $\mathbf{s} = \arg\min_{\mathbf{s}' \in \mathcal{M}} \langle \mathbf{s}', \nabla f(\boldsymbol{\alpha}^{(k)}) \rangle$ for optimization problem (4).*

Proof. Over the product domain $\mathcal{M} = \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$, the minimization $\min_{\mathbf{s}' \in \mathcal{M}} \langle \mathbf{s}', \nabla f(\boldsymbol{\alpha}) \rangle$ decomposes as $\sum_i \min_{\mathbf{s}_i \in \Delta_{|\mathcal{Y}_i|}} \langle \mathbf{s}_i, \nabla_i f(\boldsymbol{\alpha}) \rangle$. The minimization of a linear function over the simplex reduces to a search over its corners – in this case, it amounts for each i to find the minimal component of $-H_i(\mathbf{y}; \mathbf{w})$ over $\mathbf{y} \in \mathcal{Y}_i$, i.e. solving the loss-augmented decoding problem as used in Algorithm 3 to construct the domain vertex \mathbf{s} . To see this, note that for our choice of primal variables $\mathbf{w} = A\boldsymbol{\alpha}$, the gradient of the dual objective, $\nabla f(\boldsymbol{\alpha}) = \lambda A^T A\boldsymbol{\alpha} - \mathbf{b}$, writes as $\lambda A^T \mathbf{w} - \mathbf{b}$. This vector is precisely the loss-augmented decoding function $-\frac{1}{n} H_i(\mathbf{y}; \mathbf{w})$, for $i \in [n]$, $\mathbf{y} \in \mathcal{Y}_i$, as defined in (2). \square

³This choice of the radius R then gives $\frac{1}{\lambda n} R = \max_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \left\| \frac{1}{\lambda n} \psi_i(\mathbf{y}) \right\|_2 = \max_{i \in [n], \mathbf{y} \in \mathcal{Y}_i} \|A_{(i, \mathbf{y})}\|$.

D.2. Convergence Analysis

Convergence of the Batch Frank-Wolfe Algorithm 3 on the Structural SVM Dual

Theorem' 2 (Batch FW SVM). *Algorithm 3 obtains an ε -approximate solution to the structural SVM dual problem (4) and duality gap $g(\alpha^{(k)}) \leq \varepsilon$ after at most $O\left(\frac{R^2}{\lambda\varepsilon}\right)$ iterations, where each iteration costs n oracle calls.*

Proof. We apply the known convergence results for the standard Frank-Wolfe Algorithm 1, as given e.g. in Frank and Wolfe (1956); Dunn and Harshbarger (1978); Jaggi (2011), or as given at the end of the proof of Theorem 7: For each $k \geq 1$, the iterate $\alpha^{(k)}$ of Algorithm 1 (either using the predefined step-sizes, or using line-search) satisfies $E[f(\alpha^{(k)})] - f(\alpha^*) \leq \frac{4C_f}{k+2}$, where $\alpha^* \in \mathcal{M}$ is an optimal solution to problem (4).

Furthermore, if Algorithm 1 is run for $K \geq 2$ iterations, then it has an iterate $\alpha^{(\hat{k})}$, $1 \leq \hat{k} \leq K$, with duality gap bounded by $E[g(\alpha^{(\hat{k})})] \leq \frac{12C_f}{K}$, as shown e.g. in Jaggi (2011), or also in our proof for the generalized product variant provided in Appendix E, when the number of factors is set to one.

Now for the SVM problem and the equivalent Algorithm 3, the claim follows from the curvature bound $C_f \leq \frac{R^2}{\lambda}$ for the dual structural SVM objective function (4) over the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$, as given in the above Lemma 4 \square

Convergence of the Stochastic Frank-Wolfe Algorithm 4 on the Structural SVM Dual

Theorem' 3 (Stochastic FW SVM). *If $L_{\max} \leq \frac{2R^2}{\lambda n}$, then Algorithm 4 obtains an ε -approximate solution to the structural SVM dual problem (4) and expected duality gap $E[g(\alpha^{(k)})] \leq \varepsilon$ after at most $O\left(\frac{R^2}{\lambda\varepsilon}\right)$ iterations, where each iteration costs a single oracle call.*

If $L_{\max} > \frac{2R^2}{\lambda n}$, then the line-search variant of Algorithm 4 will require an additional (constant in ε) number of $2n \log\left(\frac{\lambda n L_{\max}}{2R^2}\right)$ steps to get the same error and duality gap guarantees, whereas the predefined step-size variant will require an additional $O\left(\frac{nL_{\max}}{\varepsilon}\right)$ steps.

Proof. Writing $h_0 = f(\alpha^{(0)}) - f(\alpha^*)$ for the error at the starting point used by the algorithm, the convergence Theorem 1 states that if $k \geq 0$ and $k \geq \frac{2n}{\varepsilon}(2C_f^{\text{prod}} + h_0)$, then the expected error is $E[f(\alpha^{(k)})] - f(\alpha^*) \leq \varepsilon$ and analogously for the expected duality gap. The result then follows by plugging in the curvature bound $C_f^{\text{prod}} \leq \frac{R^2}{\lambda n}$ for the dual structural SVM objective function (4) over the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$, as detailed in Appendix C (notice that it is n times smaller than the curvature C_f needed for the batch algorithm) and then bounding h_0 . To bound h_0 , we observe that by the choice of the starting point $\alpha^{(0)}$ using only the observed labels, the initial error is bounded as $h_0 \leq g(\alpha^{(0)}) = \mathbf{b}^T \mathbf{s} = \frac{1}{n} \sum_{i=1}^n \max_{\mathbf{y} \in \mathcal{Y}_i} L_i(\mathbf{y}) \leq L_{\max}$. Thus, if $L_{\max} \leq \frac{2R^2}{\lambda n}$, then we have $h_0 \leq 2C_f^{\text{prod}}$, which proves the first part of the theorem.

In the case $L_{\max} > \frac{2R^2}{\lambda n}$, then the predefined step-size variant will require an additional $\frac{2nh_0}{\varepsilon} \leq \frac{2nL_{\max}}{\varepsilon}$ steps as we couldn't use the fact that $h_0 \leq 2C_f^{\text{prod}}$. For the line-search variant, on the other hand, we can use the improved convergence Theorem 10 given in Appendix E, which shows that the algorithm require $k_0 = 2n \log\left(\frac{h_0}{2C_f^{\text{prod}}}\right)$ steps to reach the condition $h_0 \leq 2C_f^{\text{prod}}$; once this condition is satisfied, we can simply re-use Theorem 1 with k redefined as $k - k_0$ to get the final convergence rates. We also point out that the statement of Theorem 10 stays valid by replacing C_f^{prod} with any $C_f^{\text{prod}'} \geq C_f^{\text{prod}}$ in it. So plugging in $C_f^{\text{prod}'} = \frac{R^2}{\lambda n}$ and the bound $h_0 \leq L_{\max}$ in the k_0 quantity gives back the number of additional steps mentioned in the second part of the theorem statement. \square

We note that the condition $L_{\max} \leq \frac{2R^2}{\lambda n}$ is not necessarily too restrictive in the case of the structural SVM setup. In particular, the typical range of λ which is needed for a problem is around $O(1/n)$ – and so the condition becomes $L_{\max} \leq 2R^2$ which is typically satisfied when the loss function is normalized.

D.3. Implementation

We comment on three practical implementation aspects of Algorithm 4 on large structural SVM problems:

Memory For each datapoint i , our Algorithm 4 stores an additional vector $\mathbf{w}_i \in \mathbb{R}^d$ holding the contribution of its corresponding dual variables α_i to the primal vector $\mathbf{w} = A\alpha$, i.e. $\mathbf{w}_i = A\alpha_{[i]}$, where $\alpha_{[i]}$ is α_i padded with zeros so that $\alpha_{[i]} \in \mathbb{R}^m$ and $\alpha = \sum_i \alpha_{[i]}$. This means the algorithm needs more memory than the direct (or batch) Frank-Wolfe structural SVM Algorithm 3, but the additional memory is usually bounded by a constant times the size of the input data itself. In practice, in the case that the feature vectors $\psi_i(\mathbf{y})$ are sparse, we can even get the same improvement in memory requirements for \mathbf{w}_i , since $\psi_i(\mathbf{y})$ usually have the same sparsity pattern for fixed i .

Duality Gap as a Stopping Criterion Analogous as in the “classical Frank-Wolfe” structural SVM Algorithm 3 explained in Section 4, we would again like to use the duality gap $g(\alpha^{(k)}) \leq \varepsilon$ as the stopping criterion for the faster Algorithm 4. Unfortunately, since now in every step we only update a single of the many factors, such a single direction $\mathbf{s}_{(i)}$ will only determine the partial gap $g^{(i)}(\alpha^{(k)})$ in the i -th factor, but not the full information needed to know the total gap $g(\alpha^{(k)})$. Instead, to compute the total gap, a single complete (batch) pass through all datapoints as in Algorithm 3 is necessary, to obtain a full linear minimizer $\mathbf{s} \in \mathcal{M}$. For efficiency reason, we therefore only compute the duality gap every say Nn iterations for some constant $N > 1$. Then stopping as soon as $g(\alpha^{(k)}) = g(\mathbf{w}^{(k)}, \ell^{(k)}, \mathbf{w}_s, \ell_s) \leq \varepsilon$ will not affect our convergence results.

Line-Search To compute the line-search step-size for the coordinate descent Frank-Wolfe on the structural SVM, we recall that $\gamma_{opt} := \frac{\langle \alpha - \mathbf{s}, \nabla f(\alpha) \rangle}{\lambda \|A(\alpha - \mathbf{s})\|^2}$, which is valid for any $\mathbf{s} \in \mathcal{M}$. For the coordinate descent Frank-Wolfe, \mathbf{s} is equal to α for all blocks, except for the i -th block – this means that $\alpha - \mathbf{s} = \alpha_{[i]} - \mathbf{s}_{[i]}$, i.e. is zero everywhere except on the i -th block. By recalling that $\mathbf{w}_i = A\alpha_{[i]}$ is the individual contribution to \mathbf{w} from α_i which is stored during the algorithm, we see that the denominator thus becomes $\lambda \|A(\alpha - \mathbf{s})\|^2 = \lambda \|\mathbf{w}_i - \mathbf{w}_s\|^2$. The numerator is $\langle \alpha - \mathbf{s}, \nabla f(\alpha) \rangle = (\alpha - \mathbf{s})^T (\lambda A^T A \alpha - \mathbf{b}) = \lambda (\mathbf{w}_i - \mathbf{w}_s)^T \mathbf{w} - \ell_i + \ell_s$, where as before $\ell_i = \mathbf{b}^T \alpha_{[i]}$ is maintained during Algorithm 4 and so the line-search step-size can be computed efficiently. We mention in passing that when $\mathbf{s}_{(i)}$ is the exact minimizer of the linear subproblem on $\mathcal{M}^{(i)}$, then the numerator is actually a duality gap component $g^{(i)}(\alpha)$ as defined in (15) – the total duality gap is then $g^{(i)}(\alpha) = \sum_i g^{(i)}(\alpha)$ which can only be computed if we do a batch pass over all the data points, as explained in the previous paragraph.

D.4. More details on the Kernelized Algorithm

Both Algorithms 3 and 4 can be used with kernels by maintaining explicitly the sparse dual variables $\alpha^{(k)}$ instead of the primal variables $\mathbf{w}^{(k)}$. In this case, the classifier is only given implicitly as a sparse combination of the corresponding kernel functions, i.e. $\mathbf{w} = A\alpha$, where $\psi_i(\mathbf{y}) = k(\mathbf{x}_i^D, \mathbf{y}_i^D; \cdot, \cdot) - k(\mathbf{x}_i^D, \mathbf{y}; \cdot, \cdot)$ for a structured kernel $k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \rightarrow \mathbb{R}$. Note that the number of dual variables grow with the number of iterations, and so the time to take dot products also grows linearly.

Algorithm 5: Kernelized Dual Coordinate Descent Frank-Wolfe for Structural SVM

Start with $\alpha^{(0)} := (\mathbf{e}^{\mathbf{y}_1^D}, \dots, \mathbf{e}^{\mathbf{y}_n^D}) \in \mathcal{M} = \Delta_{|\mathcal{Y}_1|} \times \dots \times \Delta_{|\mathcal{Y}_n|}$

for $k = 0 \dots K$ do

 Pick $i \in_{u.a.r.} [n]$, and let $\gamma := \frac{2n}{k+2n}$

 Solve $\mathbf{y}_i := \arg\max_{\mathbf{y} \in \mathcal{Y}_i} H_i(\mathbf{y}; A\alpha^{(k)})$ (Solve the loss-augmented decoding problem (2))

$\mathbf{s}_{(i)} := \mathbf{e}^{\mathbf{y}_i} \in \mathcal{M}^{(i)}$ (having only a single non-zero entry)

 Optionally: Recalculate the best γ by line-search

 Update $\alpha_i^{(k+1)} := (1 - \gamma)\alpha_i^{(k)} + \gamma\mathbf{s}_{(i)}$

 (If line-search is used, then we also update the value $\mathbf{b}^T \alpha$ using $\mathbf{b}^T \mathbf{s}_{(i)}$)

end

E. Analysis of the Block-Coordinate Frank-Wolfe Algorithm 2

Coordinate Descent Methods Despite their simplicity and very early appearance in the literature, surprisingly few results were known on the convergence (and convergence rates in particular) of coordinate descent type methods. Recently, the interest in these methods has grown again due to their good scalability

to very large scale problems as e.g. in machine learning, and also sparked new theoretical results such as (Nesterov, 2010).

Constrained Convex Optimization over Product Domains We consider the general constrained convex optimization problem

$$\min_{x \in \mathcal{M}} f(x) \quad (14)$$

over a Cartesian product domain $\mathcal{M} = \mathcal{M}^{(1)} \times \dots \times \mathcal{M}^{(n)} \subseteq \mathbb{R}^m$, where each factor $\mathcal{M}^{(i)} \subseteq \mathbb{R}^{m_i}$ is convex and *compact*, and $\sum_{i=1}^n m_i = m$. We will write $x_{(i)} \in \mathbb{R}^{m_i}$ for the i -th block of coordinates of a vector $x \in \mathbb{R}^m$, and $x_{[i]}$ for the padding of $x_{(i)}$ with zeros so that $x_{[i]} \in \mathbb{R}^m$

Nesterov's “Huge Scale” Coordinate Descent If the objective function f is strongly smooth (i.e. has Lipschitz continuous partial gradients $\nabla_{(i)} f(x) \in \mathbb{R}^{m_i}$), then the following algorithm converges⁴ at a rate of $\frac{1}{k}$, or more precisely $\frac{n}{n+k}$, as shown in Nesterov (2010, Section 4):

Algorithm 6: Uniform Coordinate Descent Method, (Nesterov, 2010, Section 4)

```

Let  $x^{(0)} \in \mathcal{M}$ 
for  $k = 0 \dots \infty$  do
  Pick  $i \in_{u.a.r.} [n]$ 
  Compute  $s_{(i)} := \operatorname{argmin}_{s_{(i)} \in \mathcal{M}^{(i)}} \langle \nabla_{(i)} f(x^{(k)}), s_{(i)} \rangle + \frac{L_i}{2} \|s_{(i)} - x_{(i)}\|^2$ 
  Update  $x_{(i)}^{(k+1)} := x_{(i)}^{(k)} + \gamma(s_{(i)} - x_{(i)}^{(k)})$  (only affecting in the  $i$ -th coordinate block)
end

```

Using Simpler Update Steps: Frank-Wolfe / Conditional Gradient Methods In some large-scale applications, the above computation of the update direction $s_{(i)}$ can be problematic, e.g. if the Lipschitz constants L_i are unknown, or —more importantly— if the domains $\mathcal{M}^{(i)}$ are such that the quadratic term makes the subproblem for $s_{(i)}$ hard to solve.

The structural SVM is a nice example where this makes a big difference. Here, each domain factor $\mathcal{M}^{(i)}$ is a simplex of exponentially many variables, but nevertheless the linear subproblem over one such factor is often relatively easy to solve.

We would therefore like to replace the above computation of $s_{(i)}$ by a simpler one, as proposed in the following algorithm variant:

Algorithm 7: Cheaper Coordinate Descent with “Frank-Wolfe”-Type Update Steps

```

Let  $x^{(0)} \in \mathcal{M}$ 
for  $k = 0 \dots \infty$  do
  Pick  $i \in_{u.a.r.} [n]$ 
  Let  $\gamma := \frac{2n}{k+2n}$ 
  Compute  $s_{(i)} := \operatorname{argmin}_{s_{(i)} \in \mathcal{M}^{(i)}} \langle s_{(i)}, \nabla_{(i)} f(x^{(k)}) \rangle$ 
  Or alternatively, find  $s_{(i)}$  that solves this linear problem up to an additive error of  $\gamma C_f^{(i)}$ 
  Optionally: Perform line-search for the step-size:  $\gamma := \operatorname{argmin}_{\gamma \in [0,1]} f(x^{(k)} + \gamma(s_{[i]} - x_{[i]}^{(k)}))$ 
  Update  $x_{(i)}^{(k+1)} := x_{(i)}^{(k)} + \gamma(s_{(i)} - x_{(i)}^{(k)})$  (only affecting in the  $i$ -th coordinate block)
end

```

This natural coordinate descent type optimization method picks a single one of the n factors uniformly at random, and in each step leaves all other factors unchanged.

If there is only one factor ($n = 1$), then Algorithm 7 becomes the standard Frank-Wolfe (or conditional gradient descent) algorithm (Frank and Wolfe, 1956), which is known to converge⁵ at a rate of $O(1/k)$.

⁴ By additionally assuming strong convexity of f w.r.t. the ℓ_1 -norm (global on \mathcal{M} , not only on the individual factors), one can even obtain linear convergence rates, see again Nesterov (2010) and the follow-up paper Richtárik and Takáč (2011).

⁵ This convergence analysis is usually done using the fact that in each step, the primal error is reduced by a constant times the squared duality gap, see e.g. Frank and Wolfe (1956); Jaggi (2011)

Related Work In contrast to the randomized choice of coordinate which we use here, the analysis of *cyclic* coordinate descent algorithms (going through the factors sequentially) seems to be notoriously difficult, such that until today, no analysis proving a convergence rate is known. For product domains, such a cyclic analogue of our Algorithm 7 has already been proposed in Patriksson (1998), using a generalization of Frank-Wolfe iterations under the name “cost approximation”. The analysis of (Patriksson, 1998) shows asymptotic convergence, but since the method goes through the factors sequentially, no convergence rates could be proven so far.

E.1. Convergence Analysis

Decomposition of the Duality Gap The product structure of our domain has a crucial effect on the duality gap, namely that it decomposes into a sum over the n components of the domain. The Fenchel-Legendre duality gap (see Jaggi (2011, Section 2.2)) for any constrained convex problem of the above form (14), for a fixed feasible point $x \in \mathcal{M}$, is given by

$$\begin{aligned} g(x) &:= g(x; \nabla f(x)) = \max_{s \in \mathcal{M}} \langle x - s, \nabla f(x) \rangle \\ &= \sum_{i=1}^n \max_{s_{(i)} \in \mathcal{M}^{(i)}} \langle x_{(i)} - s_{(i)}, \nabla_{(i)} f(x) \rangle \\ &=: \sum_{i=1}^n g^{(i)}(x) . \end{aligned} \tag{15}$$

Curvature Also, the curvature can now be defined on the individual factors,

$$C_f^{(i)} := \sup_{\substack{x \in \mathcal{M}, s_{(i)} \in \mathcal{M}^{(i)}, \\ \gamma \in [0,1], \\ y = x + \gamma(s_{[i]} - x_{[i]})}} \frac{1}{\gamma^2} (f(y) - f(x) - \langle y_{(i)} - x_{(i)}, \nabla_{(i)} f(x) \rangle) . \tag{16}$$

We define the global product curvature as the sum of these curvatures for each factor, i.e.

$$C_f^{\text{prod}} := \sum_{i=1}^n C_f^{(i)} \tag{17}$$

Approximate Subproblems To simplify the distinction between the two variants of Algorithm 7, i.e. the exact versus the approximate solution of the linear subproblems, we use the notation $s = \text{LINEARMIN}(c, \mathcal{M}^{(i)}) := \arg\min_{s \in \mathcal{M}^{(i)}} \langle c, s \rangle$ and $s = \text{APPROXLINEARMIN}(c, \mathcal{M}^{(i)}, \varepsilon')$ if $s \in \mathcal{M}^{(i)}$ is an approximate minimizer of bounded additive error, i.e. $\langle c, s \rangle \leq \arg\min_{s \in \mathcal{M}^{(i)}} \langle c, s \rangle + \varepsilon'$.

E.2. Primal Convergence on Product Domains

The following theorem shows that after $O(\frac{1}{\varepsilon})$ many iterations, Algorithm 7 obtains an ε -approximate solution.

Theorem 7 (Primal Convergence). *For each $k \geq 0$, the iterate $x^{(k)}$ of the exact variant of Algorithm 7 (either using the predefined step-sizes, or using line-search) satisfies*

$$\mathbb{E}[f(x^{(k)})] - f(x^*) \leq \frac{2n}{k + 2n} (2C_f^{\text{prod}} + f(x^{(0)}) - f(x^*)) ,$$

where $x^* \in \mathcal{M}$ is an optimal solution to problem (14). For the approximate variant of Algorithm 7, it holds that

$$\mathbb{E}[f(x^{(k)})] - f(x^*) \leq \frac{2n}{k + 2n} (4C_f^{\text{prod}} + f(x^{(0)}) - f(x^*)) .$$

(In other words both algorithm variants deliver a solution of (expected) primal error at most ε after $O(\frac{1}{\varepsilon})$ many iterations.)

The proof of the above theorem on the convergence-rate of the primal error crucially depends on the following Lemma 8 on the improvement in each iteration.

Lemma 8. Let $\gamma \in [0, 1]$ be an arbitrary fixed step-size. Moving only within the i -th factor of the domain, we consider two variants of steps towards a direction $s_{(i)} \in \mathcal{M}^{(i)}$: Let $x_\gamma^{(k+1)} := x(\gamma)$ be the point obtained by moving towards $s_{(i)}$ using step-size γ , and let $x_{LS}^{(k+1)} := x(\gamma_{LS})$ be the corresponding point obtained by line-search, i.e. $\gamma_{LS} := \operatorname{argmin}_{\bar{\gamma} \in [0, 1]} f(x(\bar{\gamma}))$. Here for convenience we have used the notation $x(\bar{\gamma}) := x^{(k)} + \bar{\gamma}(s_{[i]} - x_{[i]}^{(k)})$ for $\bar{\gamma} \in [0, 1]$.

If the direction $s_{(i)}$ is given by $s_{(i)} := \text{LINEARMIN}(\nabla_{(i)} f(x), \mathcal{M}^{(i)})$, then in expectation over the random choice of the factor i and conditional on the value of $x^{(k)}$, it holds that

$$\mathbb{E}[f(x_{LS}^{(k+1)})] \leq \mathbb{E}[f(x_\gamma^{(k+1)})] \leq f(x^{(k)}) - \gamma \frac{1}{n} g(x^{(k)}) + \gamma^2 \frac{1}{n} C_f^{\text{prod}}.$$

If the approximate primitive $s_{(i)} := \text{APPROXLINEARMIN}(\nabla_{(i)} f(x), \mathcal{M}^{(i)}, \gamma C_f^{(i)})$ is used instead, then the above inequality holds when the constant C_f^{prod} is replaced by $2C_f^{\text{prod}}$ instead.

Proof. We write $x := x^{(k)}$, $y := x_\gamma^{(k+1)} = x + \gamma(s_{[i]} - x_{[i]})$, with $x_{[i]}$ and $s_{[i]}$ being zero everywhere except in their i -th block. We also write $d_x := \nabla_{(i)} f(x)$ to simplify the notation, and first prove the approximate case of the lemma. We use the definition (16) of the curvature constant $C_f^{(i)}$ of our convex function f over the factor $\mathcal{M}^{(i)}$, to obtain

$$\begin{aligned} f(y) &= f(x + \gamma(s_{[i]} - x_{[i]})) \\ &\leq f(x) + \gamma \langle s_{(i)} - x_{(i)}, d_x \rangle + \gamma^2 C_f^{(i)}. \end{aligned}$$

Now we use that the choice of $s_{(i)} := \text{APPROXLINEARMIN}(d_x, \mathcal{M}^{(i)}, \varepsilon')$ is a good descent direction for the linear approximation to f at x , on the i -th factor $\mathcal{M}^{(i)}$. Formally, we are given a point $s_{(i)}$ that satisfies $\langle s_{(i)}, d_x \rangle \leq \min_{y \in \mathcal{M}^{(i)}} \langle y, d_x \rangle + \varepsilon'$, or in other words we have

$$\begin{aligned} \langle s_{(i)} - x_{(i)}, d_x \rangle &\leq \min_{y \in \mathcal{M}^{(i)}} \langle y, d_x \rangle - \langle x_{(i)}, d_x \rangle + \varepsilon' \\ &= -g^{(i)}(x) + \varepsilon', \end{aligned}$$

from the definition (15) of the duality gap. Altogether, we obtain

$$\begin{aligned} f(y) &\leq f(x) + \gamma(-g^{(i)}(x) + \varepsilon') + \gamma^2 C_f^{(i)} \\ &= f(x) - \gamma g^{(i)}(x) + 2\gamma^2 C_f^{(i)}, \end{aligned}$$

the last equality following by our choice of $\varepsilon' = \gamma C_f^{(i)}$. Using that the line-search by definition must lead to an objective value at least as good as the one at the fixed γ , we therefore have shown the inequality

$$f(x_{LS}^{(k+1)}) \leq f(x_\gamma^{(k+1)}) \leq f(x^{(k)}) - \gamma g^{(i)}(x^{(k)}) + 2\gamma^2 C_f^{(i)}.$$

Finally the claimed bound on the expected improvement directly follows by taking the expectation: With respect to the (uniformly) random choice of the factor i , the expected value of the gap $g^{(i)}(x^{(k)})$ corresponding to the picked i is exactly $\frac{1}{n} g(x^{(k)})$. Also, the expected curvature of the i -th factor is $\frac{1}{n} C_f^{\text{prod}}$.

This proves the lemma for the approximate case. The analogous claims for the exact linear primitive $\text{LINEARMIN}()$ follows by the same proof for $\varepsilon' = 0$. \square

Having Lemma 8 at hand, we will now prove our above primal convergence Theorem 7 using similar ideas as for general domains, see Jaggi (2011, Section 2.3), in Clarkson (2010, Theorem 2.3).

Proof of Theorem 7. From the above Lemma 8, we know that for every inner step of the exact variant of Algorithm 7 and conditioned on the value of $x^{(k)}$, we have that $\mathbb{E}[f(x_\gamma^{(k+1)})] \leq f(x^{(k)}) - \gamma \frac{1}{n} g(x^{(k)}) + \gamma^2 \frac{1}{n} C_f^{\text{prod}}$, where the expectation is over the random choice of the factor i (this holds independently whether line-search is used or not). Writing $h(x) := f(x) - f(x^*)$ for the (unknown) primal error at any point x , this reads as

$$\begin{aligned} \mathbb{E}[h(x_\gamma^{(k+1)})] &\leq h(x^{(k)}) - \gamma \frac{1}{n} g(x^{(k)}) + \gamma^2 \frac{1}{n} C_f^{\text{prod}} \\ &\leq h(x^{(k)}) - \gamma \frac{1}{n} h(x^{(k)}) + \gamma^2 \frac{1}{n} C_f^{\text{prod}} \\ &= (1 - \frac{\gamma}{n}) h(x^{(k)}) + \gamma^2 \frac{1}{n} C_f^{\text{prod}}, \end{aligned} \tag{18}$$

where in the second line, we have used weak duality $h(x) \leq g(x)$ as explained e.g. in Jaggi (2011, Inequality (2.6)). The inequality (18) is conditional on the value of $x^{(k)}$, which is a random quantity given the previous random choices of factors to update. We get a deterministic inequality by taking the expectation of both sides with respect to the random choice of previous factors, yielding:

$$\mathbb{E}[h(x_\gamma^{(k+1)})] \leq (1 - \frac{\gamma}{n}) \mathbb{E}[h(x^{(k)})] + \gamma^2 \frac{1}{n} C_f^{\text{prod}}. \quad (19)$$

We observe that the resulting inequality (19) is of the same form as the one appearing in the standard Frank-Wolfe primal convergence proof such as in Jaggi (2011, Theorem 2.3), though with a crucial difference of the $1/n$ factor (and that we are now working with the expected values $\mathbb{E}[h(x^{(k)})]$ instead of the original $h(x^{(k)})$). We will thus follow a similar induction argument over k , but we will see that the $1/n$ factor will yield a slightly different induction base case (which for $n = 1$ can be analyzed separately to obtain a better bound). To simplify the notation, let $h_k := \mathbb{E}[h(x^{(k)})]$.

By induction, we are now going to prove that

$$h_k \leq \frac{2nC}{k+2n} \quad \text{for } k \geq 0.$$

for the choice of constant $C := 2C_f^{\text{prod}} + h_0$.

The *base-case* $k = 0$ follows immediately from the definition of C , given that $C \geq h_0$.

Now we consider the *induction step* for $k \geq 0$. Here the bound (19) for the particular choice of step-size $\gamma^{(k)} := \frac{2n}{k+2n} \in [0, 1]$ given by Algorithm 7 gives us (and also for the line-search variant given that its bound is valid for any γ):

$$\begin{aligned} h_{k+1} &\leq (1 - \frac{\gamma^{(k)}}{n}) h_k + (\gamma^{(k)})^2 \frac{C}{2n} \\ &= (1 - \frac{2}{k+2n}) h_k + (\frac{2n}{k+2n})^2 \frac{C}{2n} \\ &\leq (1 - \frac{2}{k+2n}) \frac{2nC}{k+2n} + (\frac{1}{k+2n})^2 2nC, \end{aligned}$$

where we have used that $C_f^{\text{prod}} \leq C/2$, and in the last inequality we have plugged in the induction hypothesis for h_k . Simply rearranging the terms gives

$$\begin{aligned} h_{k+1} &\leq \frac{2nC}{k+2n} \left(1 - \frac{2}{k+2n} + \frac{1}{k+2n} \right) \\ &= \frac{2nC}{k+2n} \frac{k+2n-1}{k+2n} \\ &\leq \frac{2nC}{k+2n} \frac{k+2n}{k+2n+1} \\ &= \frac{2nC}{k+1+2n}, \end{aligned}$$

which is our claimed bound for $k \geq 0$.

The analogous claim for Algorithm 7 using the approximate linear primitive APPROXLINEARMIN() follows from exactly the same argument, by replacing every occurrence of C_f^{prod} in the proof here by $2C_f^{\text{prod}}$ instead (compare to Lemma 8 also). \square

Domains Without Product Structure Our above convergence result also holds for the case of the standard Frank-Wolfe algorithm, when no product structure on the domain is assumed, i.e. for the case $n = 1$. In this case, the constant in the convergence can even be improved, since the additive term given by h_0 , i.e. the error at the starting point, will disappear: This is because already after the first step, we obtain a bound for h_1 which is independent of h_0 . More precisely, plugging $\gamma^{(0)} := 1$ in the bound (19) when $n = 1$ gives $h_1 \leq 0 + C_f^{\text{prod}} \leq C$. Using $k = 1$ as the base case for the same induction proof as above, we obtain that for $n = 1$:

$$h_k \leq \frac{2}{k+2} 2C_f^{\text{prod}} \quad \text{for all } k \geq 1,$$

which matches the convergence rate given in Jaggi (2011, Theorem 2.3). Note that in the traditional Frank-Wolfe setting, i.e. $n = 1$, our defined curvature constant becomes $C_f^{\text{prod}} = C_f$.

Dependence on h_0 We note that the only use of including h_0 in the constant $C = 2C_f^{\text{prod}} + h_0$ was to satisfy the base case in the induction proof, at $k = 0$. If from the structure of the problem we can get a guarantee that $h_0 \leq 2C_f^{\text{prod}}$, then the smaller constant $C' = 2C_f^{\text{prod}}$ will satisfy the base case and the whole proof will go through with it, without needing the extra h_0 factor. See also Theorem 10 for a better convergence result in the case where the line-search is used.

E.3. Obtaining Small Duality Gap

The following theorem shows that after $O(\frac{1}{\varepsilon})$ many iterations, Algorithm 7 will have visited a solution with ε -small duality gap in expectation:

Theorem 9 (Primal-Dual Convergence). *For each $K \geq 2$, the exact variant of Algorithm 7 (either using the predefined step-sizes, or using line-search) will yield at least one iterate $x^{(\hat{k})}$ with $1 \leq \hat{k} \leq K$ with expected duality gap bounded by*

$$\mathbb{E}[g(x^{(\hat{k})})] \leq \beta \frac{2n}{K} (2C_f^{\text{prod}} + f(x^{(0)}) - f(x^*)) ,$$

where $\beta = 3$.

The same statement holds for the approximate variant of Algorithm 7, when $2C_f^{\text{prod}}$ is replaced by $4C_f^{\text{prod}}$.

Proof. Let $K \geq 2$ be fixed. We will actually prove that the iterate of small duality gap will appear in the last half of the K iterations. To simplify notation, we will denote the expected primal and dual errors for any iteration $k \geq 0$ in the algorithm by $h^{(k)} := \mathbb{E}[h(x^{(k)})]$ and $g^{(k)} := \mathbb{E}[g(x^{(k)})]$.

By our previous primal convergence Theorem 7, we already know that the primal error satisfies $h^{(k)} = \mathbb{E}[f(x^{(k)})] - f(x^*) \leq \frac{\tilde{C}}{k+2n}$ in any iteration k , where $\tilde{C} := 2n(2C_f^{\text{prod}} + h^{(0)})$.

Between the iteration k_{\min} and K , we will now suppose that $g^{(k)}$ always stays larger than $\frac{\beta_K \tilde{C}}{k+2n}$ where $2\beta_K - 1 := \frac{K+1+2n}{K-k_{\min}+1}$. We will derive a contradiction for this assumption and then simplify the expression for better interpretability of the bound. The argument below will make clear why the quantities were defined as such. Formally, we assume that

$$g^{(k)} > \frac{\beta_K \tilde{C}}{k+2n} \quad \text{for } k \in K_{\text{set}} := \{k_{\min}, \dots, K\} .$$

For now, $k_{\min} < K$ is arbitrary, but we will see later that the tightest bound can be obtained by using $k_{\min} = \lceil \mu K \rceil$ with $\mu = 2 - \sqrt{2} \approx 0.5858$, though using $\mu = 1/2$ gives a simpler bound.

Now employing the crucial expected improvement bound from Lemma 8 for the choice of $\gamma := \frac{2n}{k+2n}$, we have $h^{(k+1)} \leq h^{(k)} - \gamma \frac{1}{n} g^{(k)} + \gamma^2 \frac{1}{n} C_f^{\text{prod}}$. It is important to observe that this bound holds both for line-search as well as for the predefined step-size γ . This gives

$$\begin{aligned} h^{(k+1)} &\leq h^{(k)} - \frac{2}{k+2n} g^{(k)} + \frac{4}{(k+2n)^2} n C_f^{\text{prod}} \\ &\leq h^{(k)} - \frac{2}{k+2n} g^{(k)} + \frac{\tilde{C}}{(k+2n)^2} . \end{aligned}$$

Plugging in our assumption that the duality gap is still “large” for $k \in K_{\text{set}}$, we obtain

$$\begin{aligned} h^{(k+1)} &< h^{(k)} - \frac{2}{k+2n} \frac{\beta_K \tilde{C}}{k+2n} + \frac{\tilde{C}}{(k+2n)^2} \\ &= h^{(k)} - \frac{(2\beta_K - 1)\tilde{C}}{(k+2n)^2} . \end{aligned}$$

The crux of the proof is now to sum up this inequality over K_{set} , i.e. from $k = k_{\min}$ up to $k = K$ and show that the RHS becomes negative, yielding a contradiction. From the telescoping sum on $h^{(k)}$, we get

$$\begin{aligned} h^{(K+1)} &< h^{(k_{\min})} - (2\beta_K - 1)\tilde{C} \sum_{k=k_{\min}}^K \frac{1}{(k+2n)^2} \\ &\leq \frac{\tilde{C}}{k_{\min}+2n} - (2\beta_K - 1)\tilde{C} \sum_{k=k_{\min}}^K \frac{1}{(k+2n)^2} , \end{aligned}$$

where in the last inequality we have just used the primal convergence Theorem 7 giving $h^{(k_{\min})} \leq \frac{\tilde{C}}{k_{\min}+2n}$. We can lower bound the summand term by using the fact that for any positive decreasing integrable function f , $\sum_{k=k_{\min}}^K f(k) \geq \int_{k_{\min}}^{K+1} f(t) dt$. So, using $f(k) := \frac{1}{(k+2n)^2}$, we have that

$$\begin{aligned} \sum_{k=k_{\min}}^K \frac{1}{(k+2n)^2} &\geq \int_{k_{\min}}^{K+1} \frac{1}{(t+2n)^2} dt = \left[-\frac{1}{t+2n} \right]_{t=k_{\min}}^{t=K+1} \\ &= -\frac{1}{K+1+2n} + \frac{1}{k_{\min}+2n} = \frac{K+1-k_{\min}}{(k_{\min}+2n)(K+1+2n)} . \end{aligned}$$

Lower bounding the summand in the relevant inequality (and keeping in mind that $(2\beta_K - 1) \geq 0$ and so we are bounding it in the right direction), we get

$$\begin{aligned} h^{(K+1)} &< \frac{\tilde{C}}{k_{\min}+2n} - (2\beta_K - 1)\tilde{C} \frac{K+1-k_{\min}}{(k_{\min}+2n)(K+1+2n)} \\ &= \frac{\tilde{C}}{k_{\min}+2n} \left(1 - (2\beta_K - 1) \frac{K+1-k_{\min}}{K+1+2n} \right) = 0 , \end{aligned}$$

where the last equality arises by substituting the value for β_K that we had chosen before. We thus get $h^{(K+1)} < 0$, i.e. that the primal error becomes negative, which yields a contradiction. Our assumption on the gap is refuted, and thus we get that there exists a $\hat{k} \in K_{\text{set}}$ such that

$$g^{(\hat{k})} \leq \frac{\beta_K \tilde{C}}{\hat{k} + 2n} \leq \frac{\beta_K \tilde{C}}{k_{\min} + 2n} ,$$

where the last inequality uses the fact that $\hat{k} \geq k_{\min}$.

The rest of this proof amounts to transform this bound to make it more interpretable and choose an appropriate μ for the definition of $k_{\min} = \lceil \mu K \rceil$. By using the fact that $\mu K \leq \lceil \mu K \rceil \leq \mu K + 1$, and substituting the value of β_K from its definition, we have

$$\begin{aligned} \frac{\beta_K}{k_{\min} + 2n} &\leq \frac{1}{2} \left(1 + \frac{K+1+2n}{K-(\mu K+1)+1} \right) \frac{1}{\mu K + 2n} \\ &= \frac{1}{K} \frac{1}{2(1-\mu)} \frac{(1-\mu)K + K + 1 + 2n}{\mu K + 2n} := \frac{1}{K} \beta(\mu, K, n) . \end{aligned}$$

Our goal is now to find an upper bound for $\beta(\mu, K, n)$ as a function of μ , and find the $\mu^* \in [0, 1]$ which minimizes it. First, let's write $K = \alpha n$ (which is always possible for some $\alpha > 0$). We get

$$\beta(\mu, \alpha n, n) = \frac{1}{2(1-\mu)} \frac{(2-\mu)\alpha + 2 + \frac{1}{n}}{\mu\alpha + 2} .$$

Note that since $\mu \in [0, 1]$, this is an increasing function of α , and so we can upper bound it by letting $\alpha \rightarrow \infty$, thus getting

$$\beta(\mu, K, n) \leq \lim_{\alpha \rightarrow \infty} \beta(\mu, \alpha n, n) = \frac{2-\mu}{2\mu(1-\mu)} := \beta_\mu ,$$

where the bound holds for any positive K and n . Minimizing β_μ with respect to $\mu \in [0, 1]$ yields the optimal $\mu^* = 2 - \sqrt{2} \approx 0.5858$, which yields $\beta_{\mu^*} \approx 2.9142$. Using $\mu = 1/2$ yields the simpler $\beta_\mu = 3$ which is the one we used in the statement of our theorem and this completes the proof. Note that a more precise statement could state that, for any $\mu \in [0, 1]$, there exists a \hat{k} in $\{\lceil \mu K \rceil, \dots, K\}$ such that

$$g^{(\hat{k})} \leq \frac{\beta_\mu \tilde{C}}{K} ,$$

with β_μ defined as above.

Finally, the proof for the approximate variant of Algorithm 7 follows exactly the same scheme but using $\tilde{C} := 2n(4C_f^{\text{prod}} + h^{(0)})$ and the relevant primal convergence statement. \square

E.4. An Improved Primal Convergence for the Line-Search Case

If line-search is used, we can improve the convergence result of Theorem 7 by showing a weaker dependence on the starting condition thanks to faster progress in the starting phase of the first few iterations:

Theorem 10 (Improved Primal Convergence for Line-Search). *For each $k \geq k_0$, the iterate $x^{(k)}$ of the line-search variant of Algorithm 7 (using the exact linear subproblem) satisfies*

$$\mathbb{E} [f(x^{(k)})] - f(x^*) \leq \frac{4nC_f^{\text{prod}}}{k - k_0 + 2n}$$

where $k_0 := \max \left\{ 0, \left\lceil \log \left(\frac{h(x^{(0)})}{2C_f^{\text{prod}}} \right) / (-\log \xi_n) \right\rceil \right\}$ is the number of steps required to guarantee that $\mathbb{E} [f(x^{(k)})] - f(x^*) \leq 2C_f^{\text{prod}}$, with $x^* \in \mathcal{M}$ being an optimal solution to problem (14), and $h(x^{(0)}) := f(x^{(0)}) - f(x^*)$ is the primal error at the starting point, and $\xi_n := 1 - \frac{1}{2n} < 1$ is the geometric decrease rate of the primal error in the first phase $k < k_0$.

Proof. For the line-search case, then the expected improvement guaranteed by Lemma 8, in marginal expectation as in (19), is valid for *any* choice of $\gamma \in [0, 1]$:

$$\mathbb{E} [h(x_{LS}^{(k+1)})] \leq (1 - \frac{\gamma}{n}) \mathbb{E} [h(x^{(k)})] + \gamma^2 \frac{1}{n} C_f^{\text{prod}} . \quad (20)$$

Because the bound (20) holds for any γ , we are free to choose the one which minimizes it subject to $\gamma \in [0, 1]$, that is $\gamma^* := \min \left\{ 1, \frac{h_k}{2C_f^{\text{prod}}} \right\}$, where we have again used the identification $h_k := \mathbb{E} [h(x_{LS}^{(k)})]$. Now we distinguish two cases:

If $\gamma^* = 1$, then $h_k \geq 2C_f^{\text{prod}}$ and so (20) at $\gamma = \gamma^*$ becomes

$$\begin{aligned} h_{k+1} &\leq \left(1 - \frac{1}{n}\right) h_k + \frac{1}{n} C_f^{\text{prod}} \\ &\leq \left(1 - \frac{1}{n}\right) h_k + \frac{1}{2n} h_k \\ &= \left(1 - \frac{1}{2n}\right) h_k. \end{aligned}$$

Giving $h_k \leq (\xi_n)^k h_0 \leq B$ (where we have chosen $\xi_n := 1 - \frac{1}{2n}$), as soon as $k \geq \log_{1/\xi_n}(h_0)/B = \frac{\log(\frac{h_0}{B})}{-\log(1 - \frac{1}{2n})}$. By employing this bound for $B := 2C_f^{\text{prod}}$, we have obtained a logarithmic bound on the number of steps that fall into the first regime case here, i.e. where h_k is still “large”. Here it is crucial to note that the primal error h_k is always decreasing in each step, due to the line-search, so once we leave this regime of $h_k \geq 2C_f^{\text{prod}}$, then we will never enter it again in subsequent steps.

On the other hand, as soon as we reach a step k (e.g. when $k = k_0$) such that $\gamma^* < 1$ or equivalently $h_k < 2C_f^{\text{prod}}$, then we are always in the second phase where $\gamma^* = \frac{h_k}{2C_f^{\text{prod}}}$. Plugging this value of γ^* in (20) yields the recurrence bound:

$$h_{k+1} \leq h_k - \frac{1}{\nu} h_k^2 \quad \forall k \geq k_0 \quad (21)$$

where $\nu := 4nC_f^{\text{prod}}$, with the initial condition $h_{k_0} \leq 2C_f^{\text{prod}} = \frac{\nu}{2n}$. This is a standard recurrence inequality which appeared for example in Joachims et al. (2009, Theorem 5, see their Equation (23)) or in the appendix of (Teo et al., 2007). We can solve the recurrence (21) by following the argument of (Teo et al., 2007), where it was pointed out that since h_k is monotonically decreasing, we can upper bound h_k by the solution to the corresponding differential equations $h'(t) = -h^2(t)/\nu$, with initial condition $h(k_0) = h_{k_0}$. Integrating both sides, we get the solution $h(t) = \frac{\nu}{t - k_0 + \nu/h_{k_0}}$. Plugging in the value for h_{k_0} and since $h_k \leq h(k)$, we thus get the bound:

$$h_k \leq \frac{4nC_f^{\text{prod}}}{k - k_0 + 2n} \quad \forall k \geq k_0, \quad (22)$$

which completes the proof. \square

Note that since for $n > 0.5$ and $-\log(1 - \frac{1}{2n}) > \frac{1}{2n}$ for the natural logarithm, we get that $k_0 \leq \left\lceil 2n \log \left(\frac{h(x^{(0)})}{2C_f^{\text{prod}}} \right) \right\rceil$ and so unless $h(x^{(0)}) \leq 2C_f^{\text{prod}}$, we get a linear number of steps in n required to reach the second phase, but the dependence is logarithmic in $h(x^{(0)})$ — instead of linear in $h(x^{(0)})$ as given by our previous convergence Theorem 7 for the fixed step-size variant (in the fixed step-size variant, we would need $k_0 = \left\lceil 2n \frac{h(x^{(0)})}{2C_f^{\text{prod}}} \right\rceil$ steps to guarantee $h_{k_0} \leq 2C_f^{\text{prod}}$). Therefore, for the line-search variant of our Algorithm 7, we have obtained guaranteed ε -small error after

$$\left\lceil 2n \log \left(\frac{h(x^{(0)})}{2C_f^{\text{prod}}} \right) \right\rceil + \left\lceil \frac{4nC_f^{\text{prod}}}{\varepsilon} \right\rceil$$

iterations.

It is also interesting to point out that even though we were using the optimal step-size in the second phase of the above proof (which yielded the recurrence (21)), the second phase bound is not better than what we could have obtained by using a fixed step-size schedule of $\frac{2n}{k - k_0 + 2n}$ and following the same induction proof line as in the previous Theorem 7 (using the base case $h_{k_0} \leq 2C_f^{\text{prod}}$ and so we could let $C := 2C_f^{\text{prod}}$). This thus means that the advantage of the line-search over the fixed step-size schedule only appears in knowing when to switch from a step-size of 1 (in the first phase, when $h_k \leq 2C_f^{\text{prod}}$) to a step-size of $\frac{2n}{k - k_0 + 2n}$ (in the second phase), which unless we know the value of $f(x^*)$, we can't know in general. In the standard Frank-Wolfe case where $n = 1$, there is no difference in the rates for line-search or fixed step-size schedule as in this case we know $h_1 \leq C_f^{\text{prod}}$ as explained at the end of the proof of Theorem 7.